KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Pijus Kuzma

COMPARSINON OF CUSTOMERS LOOK-ALIKE MODELS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Vytautas Janilionis

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

COMPARISON OF CUSTOMERS LOOK-ALIKE MODELS

Baigiamasis magistro projektas

Taikomoji matematika (kodas 621G10003)

Vadovas
Doc. dr. Vytautas Janilionis

Recenzentas
Doc. dr. Kristina Šutienė

Projektą atliko
Pijus Kuzma

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Pijus Kuzma

Taikmoji matematika (621G10003)

Baigiamojo projekto „Comparison of customers look-alike models“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2015 m. birželio mėn. 8 d.

Kaunas

Patvirtinu, kad mano, **Pijaus Kuzmos,** baigiamasis darbas tema „Comparison of customers look-alike models“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena darbo dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymu nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(studento vardas ir pavardė, įrašyti ranka)	(paraša

Turinys

Kuzma, P. Comparison of customer look-alike models. Magistro baigiamasis darbas / vadovas doc. dr. Vytautas Janilionis; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas, taikomosios matematikos katedra.
Kaunas, 2015. 46 psl.

SANTRAUKA

Esant dabartinėms rinkos sąlygoms, sparčiai didėja įvairių duomenų prieinamumas, kokybė ir įvairiapusiškumas. Įvairių apklausų duomenys, sutarčių pasirašymo sąlygos, viešoji informacija bei daugybė internetinių duomenų gali būti sutelkti į vieną sistemą, kurioje šių įvairių šaltinių duomenys yra struktūrizuojami ir sisteminami taip, kad būtų galima informaciją susieti su konkrečiu asmeniu. Tokie duomenų privalumai ypač naudingi įvairių kompanijų marketingui – turėdami visapusišką informaciją apie savo klientą, kompanijos gali lengviau atsirinkti savo tikslinę auditoriją, suprasti kokie požymiai nusako esamus ir potencialius klientus, kam atitinkamas produktas yra neaktualus, o kam jis yra reikalingas. Toks personalizuotas marketingas yra vadinamas tiesioginiu marketingu (angl. *direct marketing*).

Potencialių klientų atrinkimui tiesioginiame marketinge, dažnai naudojami klasifikavimo metodai, kur priklausomas kintamasis yra dvireikšmis ir yra daug nepriklausomų kintamųjų, kurie gali būti kategoriniai, intervalų ar vardų skalės. Klasifikavimo metodai taip pat yra taikomi daugelyje kitų sričių, tokių kaip vaizdų atpažinimas, ligos, orų, poveikio nustatymo srityse ir t.t. Vieni iš populiariausių klasifikavimo metodų, kai priklausomas kintamasis yra dvireikšmis, yra logistinė regresija, dirbtiniai neuroniniai tinklai ir sprendimo medžiai [1-8], tačiau, mūsų žiniomis, iki šiol nebuvo atlikta palyginimo tarp šių metodų su marketingo duomenimis, kurių specifika – didelės duomenų imtys ir nemažai kintamųjų, kurie tarpusavyje yra dažnai susiję.

Šio darbo tikslas – sukurti ir palyginti pagrindinių komponenčių logistinės regresijos, neuroninių tinklų ir atsitiktinių miškų metodus, įvertinti jų teikiamą naudą atsižvelgiant į tai, kaip tiksliai skirtingi metodai sugeba klasifikuoti potencialius klientus iš atsitiktinės imties. Toks palyginimas leistų padaryti išvadas apie tai, koks metodas ir kokie jo parametrai geriausiai tinka analizuojamiems duomenims: palyginti, ar skirtingi metodai klasifikuoja tuos pačius įrašus kaip geriausius / blogiausius potencialius klientus. Šis palyginimas atskleidžia, ar skirtingi metodai nesuteikia visiškai skirtingos tikimybės tam pačiam įrašui.

Darbo rezultatas – dvireikšmio kintamojo klasifikavimo uždavinio su intervalų skalės nepriklausomais kintamaisiais sprendimo metodologija ir jos realizavimas. Naudojant šią metodologiją, keletas klasifikavimo algoritmų gali būti palyginami, įvertinti ir labiausiai tinkamas metodas gali būti pasirinktas, atsižvelgiant į norimą įvertinimo metriką. Taip pat, lyginant skirtingų modelių decilius, galima nuspręsti, ar skirtingi modeliai nėra prieštaraujantys vienas kitam.

0.1. Klasifikavimo algoritmai

Šiame skyriuje apžvelgsime tris skirtingus klasifikavimo algoritmus, kurie buvo lyginami šiame darbe – pagrindinių komponenčių logistinė regresija, neuroniniai tinklai bei atsitiktiniai miškai.

0.1.1. Pagrindinių komponenčių logistinė regresija

Logistinė regresija yra viena seniausių ir populiariausių klasifikavimo metodų, tačiau vienas iš esminių logistinės regresijos trūkumų – standartiniame modelyje negali būti naudojami koreliuoti nepriklausomi kintamieji, t.y. multikolinearūs duomenys turi būti pašalinti iš duomenų imties prieš konstruojant modelį. Tarpusavyje koreliuotų duomenų į logistinės regresijos modelį negalima imti kartu todėl, kad naudojant mažiausių kvadratų metodą apskaičiuojant modelio koeficientus, esant koreliuotiems kintamiesiems, koeficientų dispersija labai stipriai padidėja ir dėl to koeficientų įverčiai pasidaro labai nestabilūs.

Šį logistinės regresijos trūkumą galima panaikinti naudojant pagrindinių komponenčių analizę ir gauti įverčius naudojant tik kelias pirmas pagrindines komponentes, kurios paaiškina didžiąją dalį duomenų dispersijos ir yra tarpusavyje nekoreliuotos. Įtraukiant pagrindinių komponenčių analizę į logistinės regresijos modelį yra gaunama taip vadinama pagrindinių komponenčių logistinė regresija (angl. *principal component logistic regression*).

0.1.2. Neuroniniai tinklai

Neuroniniai tinklai – dar vienas plačiai paplitęs metodas spręsti įvairiems klasifikavimo uždaviniams [17-19]. Neuroninių tinklų metodą galima traktuoti kaip netiesinę neparametrinę regresiją. Neuroninių tinklų modelis bando atkartoti smegenyse esančių neuronų veiklos principą – esant tam tikriems įvesties parametrams, sukuriamos paslėptos jungtys, kurių kiekviena yra sujungta su įvesties parametrais skirtingais svoriais ir tos pačios paslėptos jungtys susietos su gaunamais išvesties rezultatais su skirtingais svoriais. Jungtyse tarp įvesties, išvesties ir paslėptų jungčių yra naudojamos netiesinės funkcijos, kuriomis ir bandoma nusakyti, kokio rezultato galima tikėtis prie tam tikrų parametrų. Dvireikšmio priklausomo kintamojo atveju, abiem jungtims naudojama logistinė funkcija:

$$f_1(x) = f_2(x) = \frac{1}{1 + e^{-x}}$$

Kuomet neuroninių tinklų modelis yra apmokomas, svoriai tarp jungčių, kurie yra aukščiau pateiktų lygčių parametrai, yra apskaičiuojami bandant minimizuoti kvadratinę paklaidą:

$$MSE\ (w) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Tokiu būdu, minimizuojant vidutinę kvadratinę paklaidą, randami neuroninių tinklų modelio parametrai.

Kuriant neuroninių tinklų modelį, svarbu tinkamai pasirinkti paslėptų jungčių skaičių. Šiame darbe pasirenkamas toks paslėptų jungčių skaičius, su kuriuo modelio įvertinimo metrikos (gini koeficientas, AUC ir modelio nauda) yra didžiausi.

0.1.3. Atsitiktiniai miškai

Atsitiktiniai miškai yra sprendimo medžių metodo plėtinys, kuriuo, vietoje vieno medžio modelio, naudojamas medžių rinkinys (miškas), iš kurio kuriamas klasifikavimas atsižvelgiant į bendrą visų medžių rezultatą. Atsitiktinių medžių metodas yra glaudžiai susijęs su taip vadinama *bootstrap aggregating* metodu, kuriuo sugeneruojamos pakartotinės imtys iš originalios imties tokiu būdu, kad pakartotinių imčių dydis yra toks pat, kaip ir originalios imties, tik įrašai iš originalios imties imami atsitiktiniu būdu ir su pasikartojimais, taip skirtingose imtyse kai kurie įrašai yra pasikartojantys, o kai kurių įrašų nėra visai. Tokiu būdu galima dirbtinai pasididinti apmokymo imtį ir gauti stabilesnius modelio įverčius [11-12].

0.2. Rezultatai

Šiame skyriuje trumpai pateikti ir apibendrinami trijų skirtingų klasifikavimo metodų rezultatai, vertinimo kriterijai ir parametrai, prie kurių skirtingi modeliai geriausiai klasifikuoja.

Siekiant sukurti geriausią modelį naudojant kiekvieną metodą, buvo sukurta nemažai modelių naudojant kiekvieną metodą ir keičiant modelių parametrus, Tai leido stebėti, kaip kinta įvairios modelio savybės (skaičiavimo laikas, modelio nauda, gini koeficientas), keičiant jo parametrus. Geriausi modeliai, naudojant skirtingus metodus, buvo pasiekti esant tokiems modelių parametrams:

- Pagrindinių komponenčių logistinė regresija – pirmos penkios pagrindinės komponentės
- Neuroniniai tinklai – dvi paslėptos jungtys
- Atsitiktiniai miškai – 750 medžių

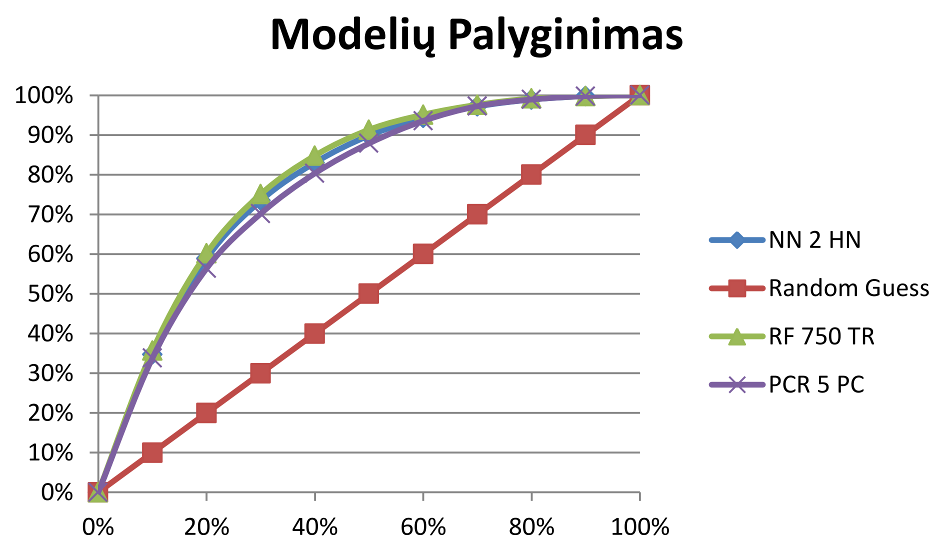
Šie parametrai pasirinkti, nes prie jų gauti didžiausi gini koeficientai ir modelių naudos.

Geriausi kiekvieno metodo modeliai taip pat buvo palyginti pagal gini koeficientą, AUC metriką bei modelio naudą. Lentelė ir grafikas žemiau apibendrina palyginimo rezultatus:

0.1 lentelė

Modelių palyginimas pagal AUC ir gini koeficientą

Model and Parameters	AUC	Gini	Rank
PCR 1-5 PCs	0,84017	0,68035	3
NN 2 Hidden Nodes	0,86071	0,72142	2
RF 750 Trees	0,86610	0,73220	1



0.1 pav. Modelių naudos kreivės

Matome (0.1 lentelė ir 0.1 pav.), kad efektyviausiai duomenis klasifikuojantis modelis yra atsitiktiniai miškai su 750 medžių, šiek tiek blogiau – neuroniniai tinklai ir pagrindinių komponenčių logistinė regresija. Naudojant atsitiktinių medžių modelį su 750 medžių, imant 30% modelio atrinktos imties, pasiekiami 75.1% klientų.

0.3. Išvados ir rekomendacijos

Šiame darbe buvo sukurti ir palyginti modeliai naudojant skirtingus klasifikavimo algoritmus – pagrindinių komponenčių logistinę regresiją, neuroninius tinklus bei atsitiktinius miškus. Naudojant kiekvieną metodą atskirai, buvo sukurta keletas modelių ir palyginimui atrinkti tie modeliai, kurių atlikimo charakteristikos (AUC, gini koeficientas, modelio pelningumas) buvo geriausios. Geriausi skirtingų metodų modeliai buvo vėl palyginti naudojant tas pačias įvertinimo metrikas ir gauta, kad turint šio uždavinio duomenis, geriausiai klasifikuojantis metodas buvo atsitiktiniai miškai su 750 medžių.

Apibendrinant darbo rezultatus, galima teigti, kad esant dideliems duomenų kiekiams ir kintamųjų skaičiui, rekomenduojama naudoti atsitiktinių miškų metodą, tačiau, esant laiko ir lėšų galimybėms, rekomenduojama naudoti pateiktą metodiką ir palyginti keletą klasifikavimo metodų, nes ši metodologija leistų objektyviau įvertinti esamą problemą ir duomenis. Kadangi apmokymo algoritmai dažnai yra priklausomi nuo duomenų, kuriais jie yra apmokomi, skirtingi metodai gali būti skirtingo efektyvumo, priklausomai nuo duomenų.

Šiame darbe pateiktos modelio įvertinimo charakteristikos yra gana plačiai naudojamos įvairių autorių [2-8] ir rekomenduojama toliau naudoti AUC (angl. *area under curve*), gini koeficientą bei modelio naudos metrikas, kurios naudingos lyginant skirtingus modelius, kurie neturi kitų bendrų modelio įvertinimo kriterijų.

Introduction

Increasing amounts of online and offline data nowadays allow companies and businesses to get a better picture of their customers – various demographic characteristics, financial status, credit risk, online behaviour, browsing preferences, etc. provide an overall view of what a customer looks like and what they might or might not to purchase. Having all this data at hand, appropriate statistical classification techniques should be used in order to be able to predict a customer.

There are many statistical techniques developed to solve the data classification problems which appear in many fields like pattern recognition, disease, weather, next item to buy prediction, etc. In the case of binary response variable, the most widely used techniques are logistic regression, artificial neural networks (ANN) and various decision trees [1-8]. However, to the best of our knowledge, there has been no comparison between such techniques and their performance in the field of marketing, where the data samples are large and the number of variables is high and the goal is to predict an individual who looks like a customer of certain company.

The main goal of this work is to compare principal component logistic regression, neural networks and random forests techniques and evaluate their performance in terms of accurate predictions of customer versus non-customer. The comparison allows drawing the conclusions about the performance of each model and make recommendations about which model and under what circumstances should be used with appropriate parameters. Another goal of this work is to investigate if the records that are assigned with the highest probabilities of each model are actually the same records. This would allow seeing if the different models are not contradicting to each other and could possible suggest combining different models to get even better classification results.

The result of this work is a methodology and its implementation for binary classification problem with continuous and binary independent variables. Using this methodology, multiple classification algorithms can be implemented and their performance evaluated and compared to each other. Depending on the actual classification problem and appropriate evaluation criteria, the best performing model can be chosen that would optimally solve the classification problem. Also, by comparing the records of each model deciles, it allows seeing if the models are predicting the same records.

1. Literature review

1.1.Previous work

Aguilera et al. [2] have used a principal component logistic regression (PCLR) model that takes only a part of principal components that would feed in the model and considers not only the variance that the principal components explain, but also the correlation between PCs and the response variable. The latter fact is very relevant to this work as the ultimate goal of customer look-alike model is to understand which variables have the biggest impact on the response variable.

Escabias et al. [6] were also using principal component analysis within logistic regression which allowed for multicollinear variables to be in a single model.

S. L. Chan et al. [3] were applying principal component regression for multiple linear regression on building projects data having relatively high number of variables. Because of high number of variables, principal component analysis was used to analyse which factors influence the project cost the most.

Kyongnam et al. [4] were comparing bagging neural network model against logistic regression model on real life data for direct marketing. They have been working with data of similar size to the data analysed in this work – c. 100,000 records with 91 variables. Authors state that bagging neural network model was superior to logistic regression.

Baesens et al. [8] were using neural network model when solving similar problem of this work – they were modelling a repeat purchase in direct marketing and were building a response model using neural network.

Lariviere et al. [5] were applying random forests both for classification and regression problems on major Belgian financial services company data which sample consisted of 100,000 records. The authors were trying to classify the data in such a way that it would be possible to predict if a customer is likely to purchase another service, if its profit is going to drop or if a customer is likely to refuse one of the products. The results of their analysis showed that applying random forests model on the data results in significantly better prediction accuracy when compared to the logistic regression model.

R. Genuer et al. [7] were investigating the variable selection problem using random forests that allows measuring the importance of each variable for the model using different metrics. Selecting a subset of model variables was also part of this work.

The above articles show only a small part of the work that has been done in solving binary classification problems. Popularity of logistic regression and neural network modelling shows that these techniques are suitable for binary classification. Promising results of random forest technique makes it interesting to see how these three modelling techniques are comparing to each other.

1.2.Classification techniques

In this section, all three classification techniques that are used throughout this work are introduced together with underlying mathematical models. In section 3.1, principal component regression is introduced, in section 3.2, details of neural network model are given and in section 3.3 random forests model is presented.

1.2.1. Principal Component Logistic regression

In this section, logistic regression, principal components analysis and principal component logistic regression are introduced. Also, disadvantages of logistic regression and advantages of principal components within logistic regression are discussed and it is shown how principal component regression can overcome some of the essential issues that classical logistic regression model has. Furthermore, selection of principle components is introduced and it's shown how principle components are usually chosen.

Logistic regression model is a type of regression, where dependent variable is binary and its probability is expressed using logistic function [2]:

$$\pi_i = \frac{e^{y(x_i)}}{1 + e^{y(x_i)}} = \frac{e^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}} \tag{1}$$

where β_i are the logistic model coefficients and x_i are the values of selected variables for the model from original observation matrix **X**. The result of the logistic regression model is the collection of coefficients β_i which are calculated using the least squares method. The estimated coefficients are then used to score the model. The model score is calculated using the following formula [2]:

$$y(x_i) = \beta_0 + \beta_1 \cdot x_{1i} + \beta_2 \cdot x_{2i} + \ldots + \beta_m \cdot x_{mi} = \beta_0 + \sum_{i=1}^{m} \beta_i x_i \tag{2}$$

And the score $y(x_i)$ is then placed in the logistic regression equation above and the result of the equation is a probability between 0 and 1. Using a specified threshold value (0.5 by default), it can be decided whether the dependent variable is 1 or 0 (in the case of this work, it can be decided whether the selected record looks like a customer or not). Logistic regression model has plenty of applications in various fields and customer classification is one of them.

1.2.2. Advantage of principal components

Another problem that arises when using the logistic regression model with high dimensional data is multicollinearity. This is also the case when using demographical data – age, gender or income are likely to influence other characteristics like credit risk, house type or value, attitude towards finance, etc. Coefficients of logistic regression are usually estimated using least square estimation. The variance of each such estimator can be expressed as [14]

$$Var\ (\beta_i) = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \sigma^2 \tag{3}$$

Where σ^2 is a variance of a noise. It can be further shown that this variance can be expressed using the eigenvalues of corresponding covariance matrix, i.e.

$$Var\ (\beta_i) = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \sigma^2 = \sigma^2 \sum_{j=1}^{i} \frac{\mathbf{v}_j \mathbf{v}_j^T}{\lambda_j} \tag{4}$$

As a result of highly correlated variables, matrix **X** is likely to become rank deficient and because of that, some of the eigenvalues of $\mathbf{X}^T \cdot \mathbf{X}$ become very close or equal to zero. By looking at the above equation, we can see that very small eigenvalues make a huge impact on the variance of least square estimator, i.e. the estimator becomes unstable. The issue of unstable coefficient estimator is the reason why highly correlated variables are not recommended to go together in a single logistic regression model if robust model coefficients are desired. However, taking only the eigenvalues that are above specified threshold could reduce the variance of the estimator.

A solution to overcome this issue is using principle components analysis (PCA) within logistic regression which is also known as principal component regression (PCR). Before starting principal component analysis, the original matrix **X** should be centred in order for the first principal component to represent the actual direction of maximum variance.

PCA transforms the centred matrix of observations $\overline{\mathbf{X}}$ of size $n \times p$ into matrix **T** of size $n \times c$ whose entries are linear combinations of $\overline{\mathbf{X}}$ and such that the entries of **T** are orthogonal to each other:

$$\mathbf{T} = \overline{\mathbf{X}} \cdot \mathbf{P} \tag{5}$$

Matrix **P** of size $p \times c$ is called principal components matrix and its column entries are called principal components that are eigenvectors of covariance matrix of centred matrix $\overline{\mathbf{X}}$ number c is chosen arbitrarily by taking a subset of eigenvectors from the covariance matrix. There are a few rules of thumb for selection of eigenvectors for the principal component matrix, but the aim is to select a subset of eigenvector that would explain the majority of the variance in the matrix $\overline{\mathbf{X}}$ **.** Using the above transformation, majority of variance of the original matrix $\overline{\mathbf{X}}$ can be explained using first few principal components.

1.2.3. Steps of principal component regression

The probability of success, or, in the case of our problem, the probability of an individual looking like a customer, can be expressed as [2]:

$$\pi_i = \frac{e^{\beta_0 + \sum_{j=1}^{p} \sum_{k=1}^{p} t_{ik} p_{jk} \beta_j}}{1 + e^{\beta_0 + \sum_{j=1}^{p} \sum_{k=1}^{p} t_{ik} p_{jk} \beta_j}} = \frac{e^{\beta_0 + \sum_{k=1}^{p} t_{ik} \gamma_k}}{1 + e^{\beta_0 + \sum_{k=1}^{p} t_{ik} \gamma_k}} \tag{6}$$

where t_{ik} are the components of transformation matrix **T**. This logistic regression model can be further expressed using matrix form:

$$L = \overline{\mathbf{X}} \cdot \beta = \mathbf{T} \cdot \mathbf{P}^T \cdot \beta = \mathbf{T} \cdot \gamma \tag{7}$$

The process of principal component logistic regression can be summarized in the following steps:

1. Take the original observation matrix **X** and centre it to get the centred matrix $\overline{\mathbf{X}}$.
2. Obtain the covariance matrix **S** of centred matrix $\overline{\mathbf{X}}$.
3. Calculate eigenvectors of the covariance matrix which will be used to form a principal component matrix and their corresponding eigenvalues.
4. Based on some criteria, select the subset of eigenvectors which will form principal components matrix **P**.
5. Calculate the transformation matrix **T** by multiplying centred observation matrix $\overline{\mathbf{X}}$ with principal components matrix **P**, i.e. $\mathbf{T} = \overline{\mathbf{X}} \cdot \mathbf{P}$.
6. Fit the logistic regression model using matrix **T** and corresponding response variable and obtain regression coefficients γ .
7. Transform the regression coefficients γ into the final regression coefficients of the original observations β using principal components matrix, i.e. $\beta = \mathbf{P} \cdot \gamma$

8. Use the final regression coefficients to get the probability of the principal component logistic regression model.

Aguilera et al. [2] has proposed a PCLR (Principal Component Logistic Regression) model that takes a reduced set of principal components of the original predictors rather than the full set of principal components. The author argues that this model improves the estimation of original parameters when compared to principal component regression.

1.2.4. Selection of PCs

There are two main criteria when selecting PCs that would feed into the model:

1. Selecting PCs that explains the largest amount of variance (having largest eigenvalues)
2. Selecting PCs that are the most correlated with the response variable

Even though some PCs might have small variances, they can be highly correlated with the response variable and vice versa – PCs explaining large amount of variance might have relatively low correlation with response variable which means that in some cases PCs having lower variance actually have a better predictive ability than those with higher variance. The bottom line of building a customer look-alike model is to be able correctly predict an individual who is likely to be a customer, therefore predictive ability should be considered as an important factor when selecting the PCs for the model.

1.3.Neural network

There has been no robust validation that neural networks model always outperforms simple logistic regression model. Some authors (Bounds et al. [17], Moutinho et al. [18]) have shown that neural networks are superior to logistic regression. However, Suh et al. [19] argued that neural network was not performing better than logistic regression. One of the reasons of fluctuations of neural networks model performance is that its performance depends on its complexity, i.e. the number of synapses or parameters (nodes), and therefore building an efficient neural network model requires additional consideration to decide whether this model is relevant to the given problem. This means that while complex neural network model might not be a good choice for a simple data and straightforward problem (the model can be over-fitted), it is likely to show its supremacy when complex data is at hand and sophisticated problem needs to be solved.

When neural network model has high level of complexity, it is very likely to have a large variance. On the other hand, low complexity model will likely have a large bias. The result of a large bias or a large variance is a classification error. It turns out that sophisticated model has a large variance, but low bias and vice versa – low complexity model has small variance and a large bias. The trade-off between variance and bias when selecting model complexity has been referred to as “Bias/Variance Dilemma” raised by Geman et al. [20]. However, it is not a straightforward task to decide what level of variance and bias should be tolerated in general – this trade-off is rather case-specific and depends on the individual problem.

1.3.1. Structure of neural network

Neural network model can be treated as non-parametric and non-linear regression model; however it was inspired by the principles of the neuron and synapses in the human brain. There are a number of different types of neural networks, but in this case we use and present the most popular neural network which is composed of three main components: input layer, hidden layer(s) and an output layer. Such a neural network is also called a multi-layer perceptron (MLP). Each layer is composed of several nodes (corresponding to neurons in human brain) and each node is connected with every node from the subsequent layer (corresponding to a synapse in human brain) and each connection has associated weight that represents the strength of the connection between two nodes. To get the value of the response variable in the output layer, the network is using two functions – one is calculated as a result of connection between input and hidden layer and the other is a result of connection between hidden and output layer:

$$z_j = f_1\left(\sum_{i=1}^{n} w_{ji} x_i\right)$$
$$y_k = f_2\left(\sum_{j=1}^{m} w_{kj} z_j\right) \tag{8}$$

x_i in the above equation represents the *i*th input variable, z_j is the *j*th node in the hidden layer and the y_k is the *k*th output value. The weight between the *i*th node and the *j*th node is denoted as w_{ji} and these weights are obtained as a result data training. Function f_1 and f_2 are the non-linear transfer functions of hidden and output layers that are chosen arbitrarily. In case of binary response variable, logistic transfer function is commonly used and in this case, both functions f_1 and f_2 will be chosen to be the same:

$$f_1(x) = f_2(x) = \frac{1}{1+e^{-x}} \tag{9}$$

The schematic drawing of neural network is depicted below:

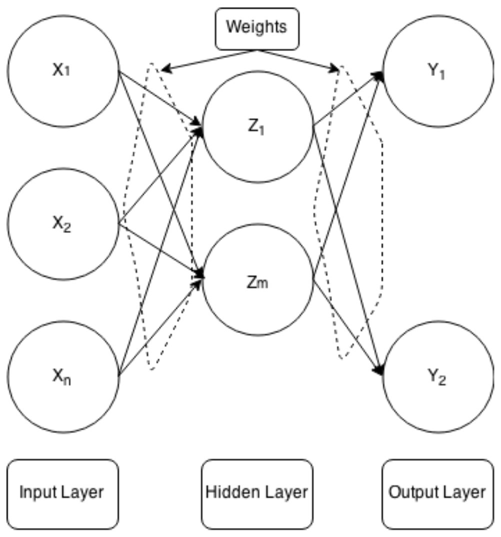


Figure 1. Structure of Neural Network

In the step of neural network training, weights of connections between the nodes are found in a way that the estimated output $\hat{y}_i$ is as close as possible to the actual output value y_i (node in the output layer). This is done by initializing the weight vector randomly and by updating it such that the mean squared error (MSE) is minimized [13]:

$$MSE(w) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{10}$$

Other error measures might also be chosen as well, such as sum of squared errors, Euclidean distance, etc. depending on specific problem or on the author’s choice.

As mentioned earlier, complexity of neural network is determined by the number of connections between the nodes and as a result of that, complexity can be described by the number of nodes in the hidden layer. In case of large number of connections (complex network), the neural network might become over-fitted and its corresponding model will rather memorise the training data than its underlying patterns. This would result in nearly perfect performance on the training data, but would fail to classify good enough for the data that was not seen before. On the other hand, a network with low number of connections might even not represent the training data well which is a first indication of poor model performance Therefore, in order to optimize the network, according to [4], the number of nodes in the hidden layers having the smallest prediction error is chosen. In case of this work, the model having the largest AUC measure and gini coefficient will be selected.

1.4.Random forests

Before explaining the random forests model, bootstrap aggregating technique will be introduced that forms a part of the idea behind the random forests model.

1.4.1. Bagging

One of the options to reduce the model variance without increasing bias, is to use bootstrap aggregating that is abbreviated simply as *bagging* which was first proposed by Breiman in 1996 [12]. The idea of bagging is that if one has multiple training data sets of the same size *n* and the same distribution, then one can get better and more robust predictors than using a single training data set of size *n*. The problem is that in real life, it is not always possible to get multiple data sets to be used for training. However, to imitate the process of generating multiple training data sets, Breiman has introduced bagging procedure that repeatedly takes bootstrap samples $\{S^B\}$ of size *n* from the original training data set **X** of size *n*. Resulting bootstrap samples $\{S^B\}$ are forming replicated data sets and each of those data sets consists of *n* records that are sampled randomly with replacement. As a result of sampling with replacement, some records from original training data set **X** might appear multiple times in the bootstrap sample while some records might not appear in the sample at all. Actually, about 67% of the original training data are selected within each bootstrap sample.

Once bootstrap samples are formed, each sample then is used to train random forest model which results in m forest (therefore, the name of the method is called random forests). To aggregate the results of m forests, voting process is used, i.e. the output that appears the most times is selected for the model.

1.4.2. Random forests model

Random forest can be treated as an ensemble method and it is an extension of decision trees (widely used in marketing for segmentation) technique especially useful for binary classification problem. Random forests were first proposed by L. Breiman [11] in 2001 and its popularity is increasing since then, mainly because of its simple structure and better results compared to other classification techniques. The idea behind random forest is that instead of growing a single tree, a collection of trees is grown instead and for each tree a different data sample of the same size is used which is obtained by bootstrap aggregating that was introduced in the section above. Furthermore, each tree is grown using a random set of independent variable of size *m* that are used to split the nodes. The number *m* should be selected such that it would be significantly smaller than the total number *n* of predictor variables available. According to Breiman, *m* should be a square of *n*, i.e. $m \approx \sqrt{n}$.

Steps of random forests:

1. Create *b* bootstrap samples consisting of *n* records from the training set **X** which also consists of *n* records.
2. For each of the bootstrap sample, randomly sample *m* ($m \approx \sqrt{n}$) independent variables from the full list of independent variables and grow a single tree using only *m* random variables to split the nodes.
3. Aggregate the predictions of all *b* trees into a single prediction using voting or averaging.

Important advantages of random forests is that they provide more robust results compared to decision trees, especially in terms of outliers and noise; they do not overfit because of the Law of Large Numbers for prediction error, which, as number of trees in the forest is increasing, converges to a certain value.

2. Research methodology

2.1. Data preparation

In this work, the data is a sample of 38,698 records from one of the European credit card providers and 2,246,873 random sample of adult population from the corresponding country. Each record has 262 variables associated with it that are mainly geodemographic variables like age, gender, income, credit risk, preferences, etc. The full list of the variables with their description is given in the appendix. The problem that is addressed in this work is to create a customer look-alike model that would be able to predict accurately if a given record (person) looks like a customer or not. Therefore, the response variable is a binary variable and gets the value of 1 if a record belongs to a customer class and 0 otherwise.

As this is the real world data, some data preparation will be involved in order to have meaningful variables and their values.

To begin with, the list of the available independent variables was reduced. There are two main reasons for this:

1. Some of the variables' categories are not significantly discriminative with respect to the dependent variable, i.e. the change in some variables value doesn't make any influence on dependent variable.
2. Some of the variables are not relevant to the problem of predicting a customer for financial product, e.g. variables like newspaper preference or ownership of a DVD

player are not appropriate indicators of a customer of such type. This variable selection can also be referred as business sense which takes into account not only statistical properties of the variable, but also how it is working in the real world.

Variable significance with respect to the dependent variable was carried out by making a simple profiling analysis – distribution of each variable of the customer file and the random sample with respect to the dependent variable were compared and significant values of variable were identified by using a formula:

$$Index = \frac{\% \ of \ \ customer \ \ \ file}{\% \ of \ \ sample \ \ \ file} \times 100 \qquad (11)$$

Index of 120 and above or 80 and below identifies if the values of a variable are over represented or under represented for a given variable. An example of profile variable *Age Band* with its corresponding index values is shown below:

Age Band	Customer	Country Base	Total	Customer	Country Base	Total	Customer vs. Country Base Index
Unknown	0	330	330	0,0%	0,0%	0,0%	0
16-20	303	95.294	95.597	0,8%	4,2%	4,2%	18
21-24	970	117.971	118.941	2,5%	5,3%	5,2%	48
25-29	2.546	178.834	181.380	6,6%	8,0%	7,9%	83
30-34	5.975	194.664	200.639	15,4%	8,7%	8,8%	**178**
35-39	6.425	185.246	191.671	16,6%	8,2%	8,4%	**201**
40-44	6.870	206.400	213.270	17,8%	9,2%	9,3%	**193**
45-49	6.248	221.952	228.200	16,1%	9,9%	10,0%	**163**
50-54	4.594	213.205	217.799	11,9%	9,5%	9,5%	**125**
55-59	2.551	177.745	180.296	6,6%	7,9%	7,9%	83
60-64	1.214	156.014	157.228	3,1%	6,9%	6,9%	45
65-69	597	155.388	155.985	1,5%	6,9%	6,8%	22
70-74	216	113.891	114.107	0,6%	5,1%	5,0%	11
75+	189	229.939	230.128	0,5%	10,2%	10,1%	5
TOTAL	**38.698**	**2.246.873**	**2.285.571**	**100,0%**	**100,0%**	**100,0%**	**100**

Table 1. Example of index value for variable *Age Band*

From the above table it can be seen which age bands are the best indicators of a record being a customer, i.e. records aged 30-54 are significantly more likely to be customers than the records aged 16-29 and 55+ when compared to the country base.

Another step for data preparation is removing records with missing variable values – in that way the models will only be trained on the information that is known. This also prevents from making any decisions about an individual if some of its characteristics are unknown.

After these data preparation steps were applied, it was also decided to have the population consisting of 20% of customers and 80% of random sample. As it was not possible to increase the customer population, random sample population was decreased resulting in 36,757 customer records, 147,028 random sample records and 39 variables that were left after variable reduction step applied and this was the data on which the models were trained and validated. The list of 39 variables that were selected for modelling is given below with the corresponding description of each variable:

Variable No.	Variable	Variable Description	Variable Type
1	MaritalStatus_COH	Marital status of *cohabitting*	Indicator
2	FB_Investments_E	Individuals with credit difficulties	Indicator
3	FB_Attitude_D_E	Individuals that are less likely to save money	Indicator
4	FB_ChannelPref_A_D	Individuals that prefer online channel for communication	Indicator
5	CreditCardTransfers_Y	Individuals that use credit card transfers	Indicator
6	HaveHomeCollectedCredit_Y	Individuals that have had home collected credit	Indicator
7	Loan_Y	Individuals that have taken loan	Indicator
8	earlyadopter_Y	Individuals that are more liekly to adopt quickly	Indicator
9	CSP_svBADPUB_Y	Individuals that have bad public debt at individual level	Indicator
10	CSP_svCCJLAST_Y	Individuals that have had financial court in last 6years at individual level	Indicator
11	CSP_svsBADPUB_Y	Individuals that have bad public debt at surname level	Indicator
12	CSP_svSCCJLAST6_Y	Individuals that have had financial court in last 6years at surname level	Indicator
13	Tenure_SR	Individuals that live in social rent households	Indicator
14	HouseType_T_F	Individuals that live in terrace houses or flats	Indicator

15	FirstTimeBuyer_mdl_Y	Individuals that have taken their first mortgage	Indicator
16	ChildPresent_mdl_Y	Presence of children	Indicator
17	ISA_mdl_N	Presence of individual savings account	Indicator
18	AgeBandDesc_mdl_W	Age band	Numeric
19	LengthOfResidency_W	Length of residency	Numeric
20	FB_Credit_mdl_W	Credit behaviour	Numeric
21	FB_Savings_mdl_W	Savings behaviour	Numeric
22	FB_Attitude_mdl_W	Attitude towards finance	Numeric
23	CreditCardTransfers_mdl_pc_W	Likelihood of credit card transfers	Numeric
24	HaveHomeCollectedCredit_mdl_pc_W	Likelihood of home collected credit	Numeric
25	Loan_mdl_pc_W	Likelihood of loan	Numeric
26	PayCreditFull_mdl_pc_W	Likelihood of paying credit in full	Numeric
27	ConnectedGroup_W	Connected group	Numeric
28	CouncilTaxBand_W	Council tax band	Numeric
29	HouseholdLengthOfResidency_W	Household length of residency	Numeric
30	HouseholdIncome_mdl_W	Household income	Numeric
31	SocialClass_mdl_W	Social class	Numeric
32	CAMEO_UK_Group_W	Group according to affluence	Numeric
33	CAMEO_Financial_Group_W	Group according to financial status	Numeric
34	CAMEO_Unemployment_W	Group according to unemployment	Numeric
35	nGauge_P_CRisk_Public_bd_m01_W	Public credit risk	Numeric
36	GR_a_n_ccj_i_W	No. Of individuals having financial court within a postcode	Numeric
37	GR_adult_1_i_W		Numeric
38	GR_allh_a_v_i_W		Numeric
39	GR_ccj_1_i_W	No. Of individuals having one financial court within a postcode	Numeric

Table 2. List of variables selected for modelling

When solving classification problem, one of the drawbacks in the data is that data often contains categorical variables that need to be transformed into numeric variables in the appropriate manner. To handle the categorical data, the weight of evidence (WoE) transformation is used throughout this work. Weights of evidence are calculated using the following formula:

$$WoE = \ln\left(\frac{\%\ of\ \ customers}{\%\ of\ \ sample}\right) \tag{12}$$

The advantage of using weights of evidence is that they not only transform the categorical data to numerical (interval scale), but it also takes into account how each variable category compares with the sample data with respect to the dependent variable. An example of variable *Gender* that was transformed using weight of evidence is given below:

Gender	Customer	Country Base	Total	Customer	Country	Total	Weight Of Evidence
Female	19.429	1.061.098	1.080.527	50,2%	47,2%	47,3%	0,06
Male	17.260	1.020.090	1.037.350	44,6%	45,4%	45,4%	-0,02
Unknown	2.009	165.685	167.694	5,2%	7,4%	7,3%	-0,35
TOTAL	**38.698**	**2.246.873**	**2.285.571**	**100,0%**	**100,0%**	**100,0%**	

Table 3. Example of Weight of Evidence for *Gender* variable

2.1. Training and validation data

In order to be able to validate each model created, the original data set was divided into training and validation data sets. Training and validation data sets were split randomly and training data set contained 70% of the original data whereas validation data set contained 30% of the original data. The distribution of customers among both data sets can be seen in the tables below:

Customer	Volume	Percentage
0	103,299	80.04%
1	25,678	19.96%
Total	**128,977**	**100%**

Table 4. Summary of training data set

Customer	Volume	Percentage
0	43,729	79.92%
1	10,989	20.08%
Total	**54,718**	**100%**

Table 5. Summary of validation data set

Training data was used for model development and after each model was completed, validation data was used to validate the developed model and to see its performance.

2.2. Comparing the models

The techniques that that are compared in this work don’t have many statistical measures of model goodness and performance in common, therefore we stick to the following measures of model goodness – area under ROC curve (AUC), gini coefficient and model gain. AUC is a very common measure to evaluate model performance [4, 5, 8]. In order to calculate AUC, one first has to draw the ROC curve which is obtained by generating a number of confusion matrices with different threshold values for the probability of success of the event and corresponding true positive rate (TPR) and false positive rate (FPR) are taken. Given the confusion matrix:

$$\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} \tag{13}$$

The corresponding TPR and FPR are calculated using the following formulas:

$$TPR = \frac{TP}{(TP + FN)} \tag{14}$$

$$FPR = \frac{FP}{(FP + TN)} \tag{15}$$

Each confusion matrix generates one point of the ROC curve – FPR stands for *x* coordinate and TPR stands for *y* coordinate in the ROC curve. After generating a number of confusion matrices, ROC curve of corresponding model is obtained and then it is possible to calculate the AUC by numerically integrating the area under the curve. The AUC is calculated using the formula:

$$AUC = \sum_{i=1}^{v-1} \frac{(y_i + y_{i+1}) \cdot \Delta x}{2} \tag{16}$$

where y_i represents the true positive rate and Δx represents the change of false positive rate on the *x* axis. Both y_i and Δx are in the interval between 0 and 1. Once the AUC is calculated, it can be used to calculate another model goodness measure – gini coefficient, which is calculated using the following formula [1]:

$$Gini = 2 \times AUC - 1 \tag{17}$$

Model gain, also known as lift curve, is another measure of model performance [15]. While AUC and gini coefficient are statistical properties of the model, model gain shows the actual benefit of the model compared to random guess. In order to calculate the model gain, all scored model records are ordered in descending order of model probability and then all records are divided into 10 deciles. For each decile, percentage of total customers is calculated and for a good model it is expected that percentage of customers is significantly higher in the top deciles compared to lower ones. An example of model gain can be seen in the result section.

A combination of these two types of model assessment is used to select the model. Usually the model having the highest AUC (therefore, gini coefficient as well) and having the best model gain is chosen.

2.3. Software

For the implementation of the methodology that solves the binary classification problem and compares the different models, two different software packages were used – R and SAS. Wide variety of statistical packages that are available in R gave a good choice of tools that could be used when creating different models. On the other hand, SAS is convenient software for data manipulation and structuring tasks, also it provides reasonable plotting features. Therefore, R software was used for the development of all models using three different classification techniques and SAS software was used to summarise the results, produce the gain charts, ROC curves, compare the models and their deciled selections. The results from R were exported and then imported to SAS which were then used to produce the majority of the outputs that are depicted in the sections below. Essential parts of R and SAS code can be found in the appendix.

3. Research results and conclusions

In this section the results of each modelling technique are presented. As described earlier, the aim of the comparison is to see the same performance measures across all models in order to adequately evaluate each model's performance. Therefore, model gains, AUC and Gini coefficient will be used as a measure to evaluate how each model is performing. In addition to that, to select the best parameters for each model, specific measure for separate models will be used.

3.1.1. Principal component logistic regression results

As described in the section about principal component regression, the initial steps of the analysis are to centre the matrix, calculate the covariance matrix and then obtain corresponding eigenvalues and eigenvectors which will be used to create a principal component matrix. Once these steps are done, various combinations of principal components (eigenvectors of covariance matrix) can be chosen for building the model. The eigenvalues of the covariance matrix are depicted in the figure and table below:

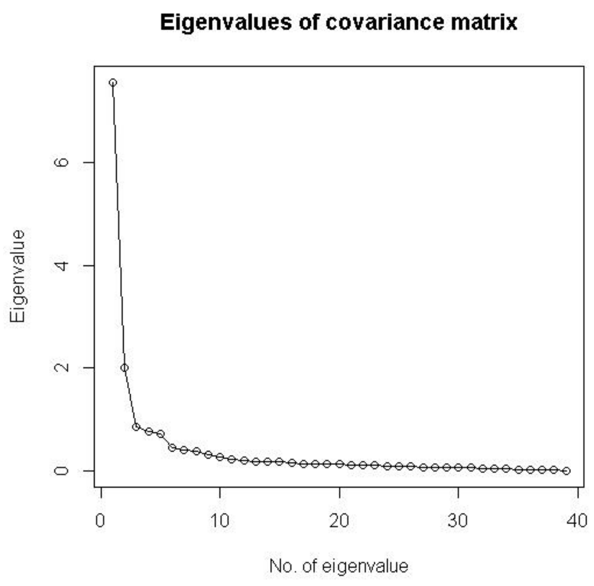


Figure 2. Eigenvalues of covariance matrix

No. Of E-value	Eigenvalue	No. Of E-value	Eigenvalue	No. Of E-value	Eigenvalue	No. Of E-value	Eigenvalue
1	7,558472	11	0,239068	21	0,123767	31	0,067977
2	2,010771	12	0,213235	22	0,114834	32	0,054073
3	0,859854	13	0,196686	23	0,112773	33	0,053101
4	0,781179	14	0,186372	24	0,102543	34	0,045288
5	0,736693	15	0,176976	25	0,097023	35	0,033115
6	0,455697	16	0,161484	26	0,090393	36	0,027270
7	0,415270	17	0,149021	27	0,079960	37	0,026615
8	0,382369	18	0,142754	28	0,078161	38	0,021114
9	0,310670	19	0,139599	29	0,073055	39	0,001742
10	0,266812	20	0,131554	30	0,070673	**Mean**	**0,430462**

Table 6. Eigenvalues of covariance matrix

For the initial selection of principal components, a rule of thumb was used to select eigenvalues whose value is above the average of all eigenvalues. From the table above, it can be seen that the average eigenvalue is 0.430462 and therefore first six eigenvectors were selected as principal components. It can also be seen from the eigenvalues plot in the figure above that a threshold for selecting principle components is right at the angle where the so called elbow starts forming and after that point the eigenvalues are getting very small values. Other combinations of principal components will be shown later.

Using the first six principle components, the following model gains were achieved:

Decile	Decile Volume	Customers	Customers %	Customers Cumulative %
1	5472	3715	33,8%	33,8%
2	5472	2481	22,6%	56,4%
3	5472	1517	13,8%	70,2%
4	5472	1118	10,2%	80,4%
5	5471	829	7,5%	87,9%
6	5472	619	5,6%	93,5%
7	5472	406	3,7%	97,2%
8	5472	186	1,7%	98,9%
9	5472	95	0,9%	99,8%
10	5471	23	0,2%	100,0%
Total	**54718**	**10989**	**100,0%**	**100,0%**

Table 7. Principal component logistic regression using first 6 eigenvectors as principal components

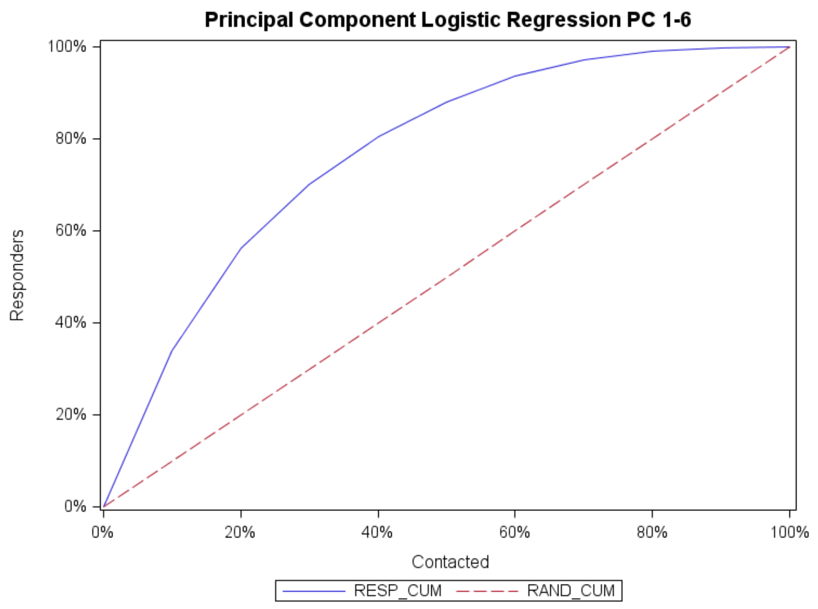


Figure 3. Model gains chart using the first six principal components

A number of models were created by selecting different amounts and combinations of principle components. An attempt was made to find out how decreasing the number of principle components in the model would affect the corresponding model performance. The results of varying principle components in the model and the resulting AUC and gini coefficient are presented in the table and figure below:

Principle Components	AUC	Gini	Change in Gini	Change in Gini %
1	0,7830	0,5660	-	-
1-2	0,7833	0,5667	0,0007	0,1%
1-3	0,8382	0,6764	0,1098	19,4%
1-4	0,8386	0,6771	0,0007	0,1%
1-5	0,8402	0,6803	0,0032	0,5%
1-6	0,8402	0,6803	0,0000	0,0%
1-7	0,8401	0,6803	-0,0001	0,0%
1-5,7	0,8401	0,6803	0,0000	0,0%

Table 8. Gini vs. number of selected principle components

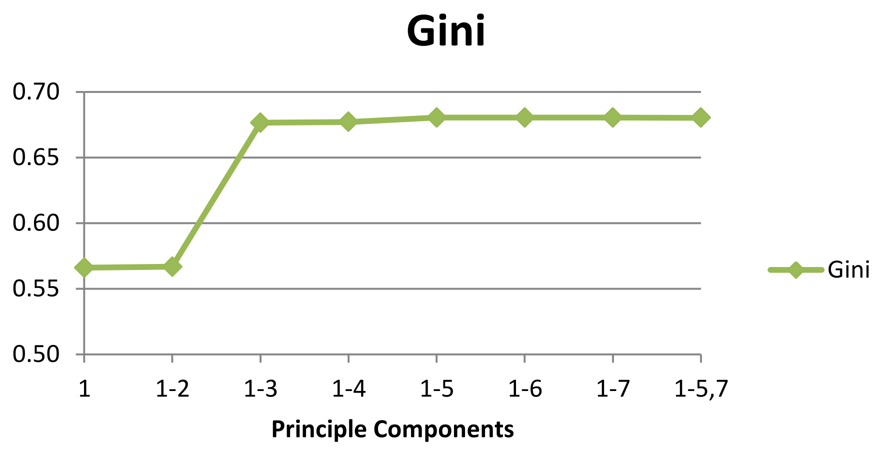


Figure 4. Gini vs. no. of principal components in principal component logistic regression model

It can be seen from the above that decreasing the number of principle components in the model from six to one by one, results in lower gini coefficient which means the poorer model performance. However, selecting more than five principle components in the model did not result in better model performance which supports the fact that first few principle components are explaining the majority of variance and the ones having smaller eigenvalues can be removed from the model.

As gini coefficient and AUC measures are related to the model gains measure, from the table above, we can draw the conclusion that for the given data, principal component logistic regression model with first five principle components selected, provides the best performance and this model is selected for the model comparison step. The model with five principal component has the following model gains:

Decile	Decile Volume	Customers	Customers %	Customers Cumulative %
1	5472	3716	33,8%	33,8%
2	5472	2480	22,6%	56,4%
3	5472	1518	13,8%	70,2%
4	5472	1115	10,1%	80,3%
5	5471	830	7,6%	87,9%
6	5472	621	5,7%	93,5%
7	5472	406	3,7%	97,2%
8	5472	185	1,7%	98,9%
9	5472	95	0,9%	99,8%
10	5471	23	0,2%	100,0%
Total	**54718**	**10989**	**100,0%**	**100,0%**

Table 9. Model gains of best model using principal component logistic regression

3.1.2. Random forests models results

A number of random forests models were created using different number of trees in each model. For each model, area under curve (AUC), Gini coefficient, confusion matrix and time taken were measured. By varying model parameters, it was possible to see how number of trees in the forest is affecting the model gains and the time taken.

Table below summarises the results of each random forests model and shows the relation between model complexity and its gain.

No. Of Trees	AUC	Gini	Change in Gini	Change in Gini %	Time Taken (seconds)
10	0,8231	0,6462	-	-	4
50	0,8576	0,7152	0,068957	10,67%	14
100	0,8621	0,7243	0,009079	1,27%	24
150	0,8632	0,7263	0,002055	0,28%	57
300	0,8649	0,7298	0,003509	0,48%	75
500	0,8659	0,7318	0,001978	0,27%	119
750	0,8661	0,7322	0,000375	0,05%	184
1000	0,8660	0,7320	-0,00016	-0,02%	236
500, 16v *	0,8574	0,7147			50

Table 10. Results of random forests models

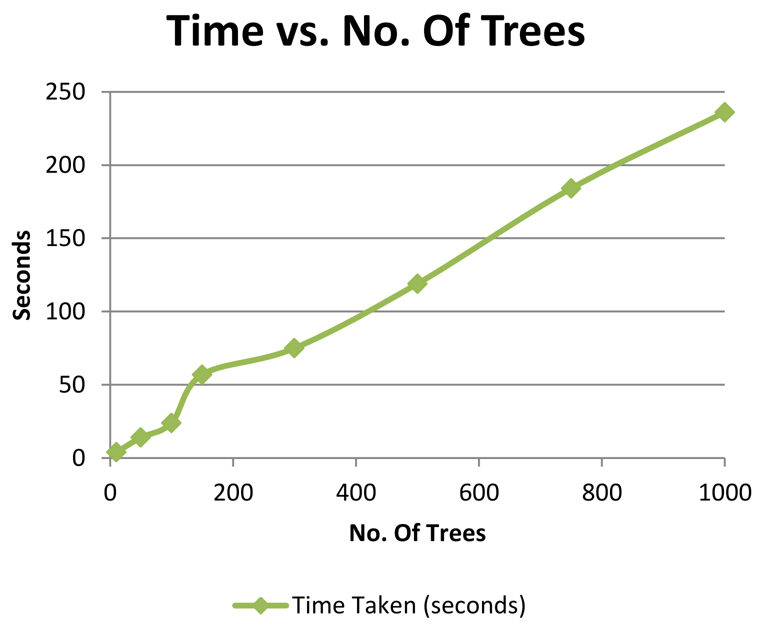


Figure 5. Time vs. no. of trees in random forest models

As it can be seen from the table and the figure above, increasing the number of trees in the random forest model results in almost linear increase in computational time. Also, increase in trees results in a better model performance, i.e. increase in AUC and Gini coefficient. However, as number of trees increases, the change of the AUC and Gini coefficient becomes very minor above 500 trees. Actually, increasing the number of trees from 500 to 1,000, Gini coefficient increases only by 0.0009 – from 0.5835 to 0.5844, which is only ~0.15%. The change of Gini coefficient as number of trees increases depicted in the figure below:

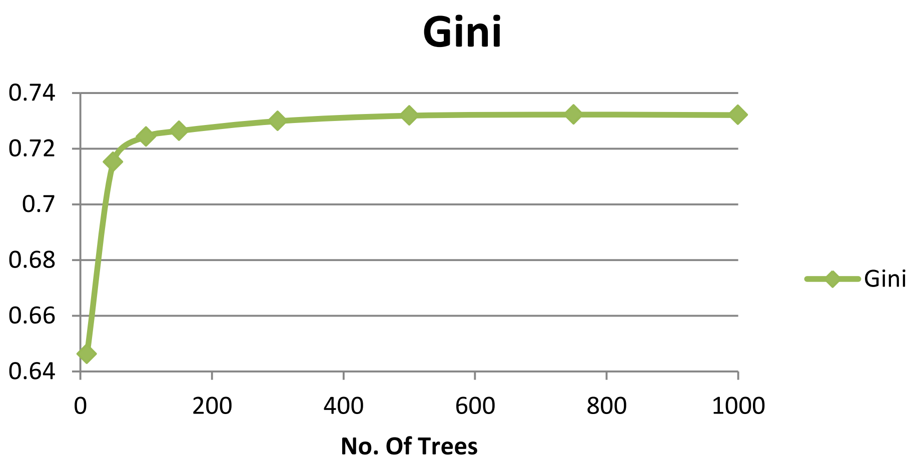


Figure 6. Gini vs. no. of trees in RF model

Taking into consideration the time taken and increase in Gini, it was decided that random forest model with 750 trees is the optimal choice and it will be taken forward for comparison with other models. Below are the gains chart and deciled model selection records:

Decile	Decile Volume	Customers	Customers %	Customers Cumulative %
1	5472	3920	35,7%	35,7%
2	5472	2694	24,5%	60,2%
3	5472	1637	14,9%	75,1%
4	5472	1068	9,7%	84,8%
5	5471	717	6,5%	91,3%
6	5472	416	3,8%	95,1%
7	5472	274	2,5%	97,6%
8	5472	173	1,6%	99,2%
9	5472	61	0,6%	99,7%
10	5471	29	0,3%	100,0%
Total	**54718**	**10989**	**100,0%**	**100,0%**

Table 11. Deciled selection of customers using RF model with 750 trees

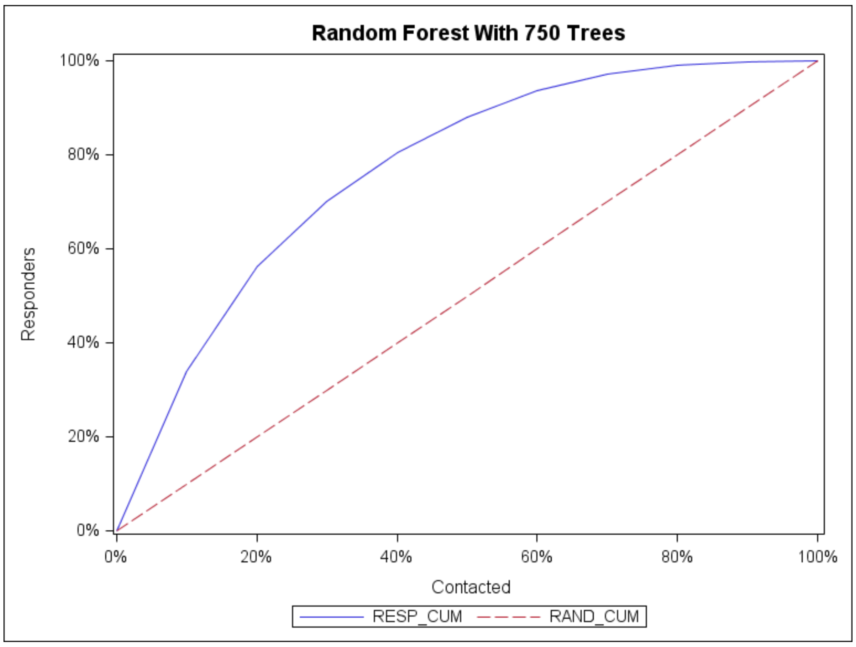


Figure 7. Model gains chart of random forest model with 750 trees

3.1.3. Variable importance in random forests models

Using random forests, it is possible to estimate, which variables have the largest impact for the model. There are two main measures that can be used to see the variable importance – mean decrease in accuracy (MDA) and mean decrease in gini (MDG). As advised by M.L.

Calle et al.[16], mean decrease in accuracy might be unstable when small perturbations occur in the dataset, whereas mean decrease in gini turns out to provide more robust results, compared to MDA. Therefore, mean decrease in Gini was used for important variable selection. MDG figure is depicted below that shows mean decrease of each variable that was used in the initial model. As random forest model with 750 trees was selected as optimal model, mean decrease in gini figure is drawn from this model.

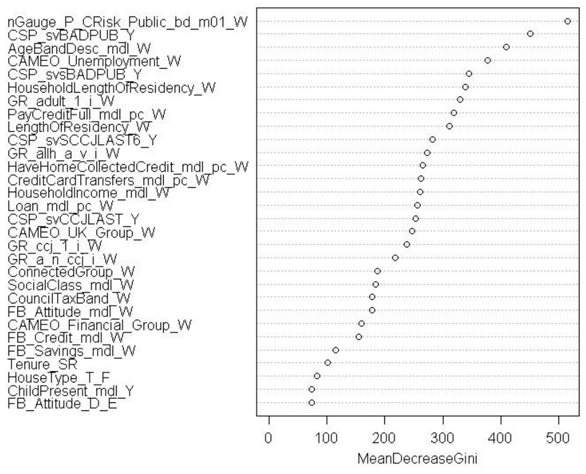


Figure 8. Mean decrease in Gini of RF model variables

From the figure above, it can be seen that nGauge, CSP_svBadPub and Age Band variables have the largest mean decrease in Gini values. These variables represent credit risk, bad public debt at individual level and age respectively. In order to see how the random forest model works on reduced list of variables, square root of all available variables were selected for a reduced list of variables, resulting in $\sqrt{262} \approx 16.2 \Rightarrow 16$ variables. These 16 variables were taken as top 16 variables from the MDE figure.

The results of the model with the most important variables can be seen in the table below and it can be seen that variable reduction from 39 to 16 resulted in Gini decrease of 0.0171 or 2.3% only.

3.1.4. Results of neural network models

Just like in the case of random forests technique, a number of models were run using neural network modelling. For each model, area under curve (AUC), Gini coefficient, confusion matrix and time taken were measured. By varying model parameters, it was possible to see how number of hidden nodes in the neural network model is affecting the model gains and the time taken.

Table below summarises the results of each neural network model and shows the relation between model complexity and its gain.

No. Of Hidden Nodes	AUC	Gini	Change in Gini	Change in Gini %	Time Taken (seconds)
1	0,85673	0,71346	-	-	16
2	0,860712	0,721425	0,007964	1,1%	55
3	0,854595	0,709191	-0,01223	-1,7%	52
5	0,849367	0,698734	-0,01046	-1,5%	172
7	0,845125	0,69025	-0,00848	-1,2%	407
10	0,836752	0,673504	-0,01675	-2,4%	675
12	0,835215	0,670429	-0,00307	-0,5%	694
15	0,829494	0,658988	-0,01144	-1,7%	1654
20	0,813838	0,627677	-0,03131	-4,8%	3372

Table 12. Neural network model gain and its complexity

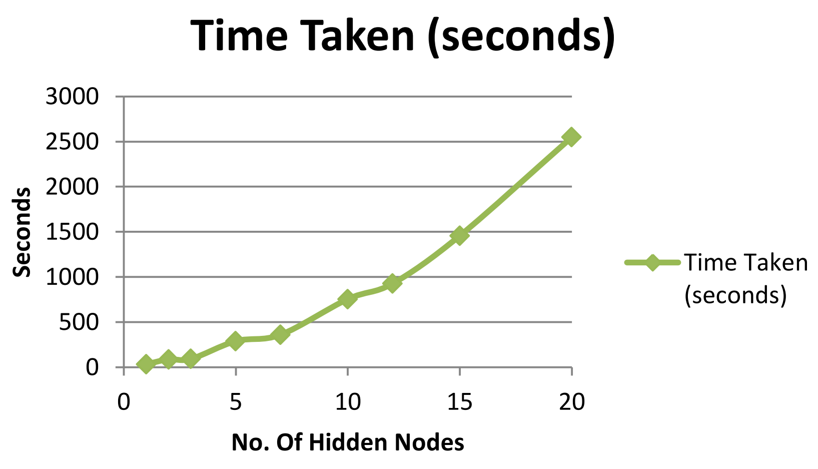


Figure 9. Time taken vs. no. of hidden nodes in the neural network models

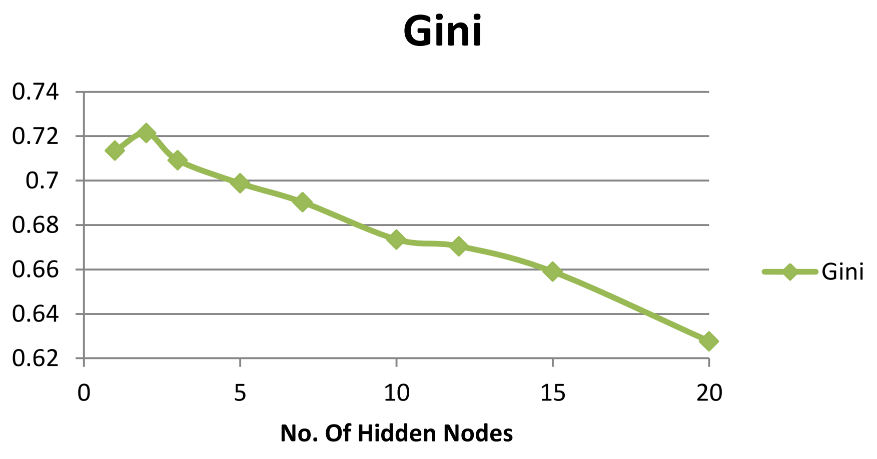


Figure 10. Number of hidden nodes vs. gini coefficient

As it can be seen from the table and the figure above, increasing the number of hidden nodes in the neural network model does not necessarily result in better model performance in terms of AUC and Gini coefficient. It can be seen that the number of hidden nodes resulting in the highest Gini coefficient is two hidden nodes in the hidden layer.

Also, as number of hidden nodes increases, time is increasing as well as can be seen in the figure above. Therefore, increasing number of hidden nodes in the neural network model with the given data does not reduce the time complexity and does not increase the model performance.

The model gains and gains chart of neural network model with two hidden nodes are depicted below:

Decile	Decile Volume	Customers	Customers %	Customers Cumulative %
1	5472	3816	34,7%	34,7%
2	5472	2658	24,2%	58,9%
3	5472	1599	14,6%	73,5%
4	5472	1052	9,6%	83,0%
5	5471	749	6,8%	89,9%
6	5472	487	4,4%	94,3%
7	5472	319	2,9%	97,2%
8	5472	190	1,7%	98,9%
9	5472	93	0,8%	99,8%
10	5471	26	0,2%	100,0%
Total	**54718**	**10989**	**100,0%**	**100,0%**

Table 13. Model gains of neural network model with two hidden nodes

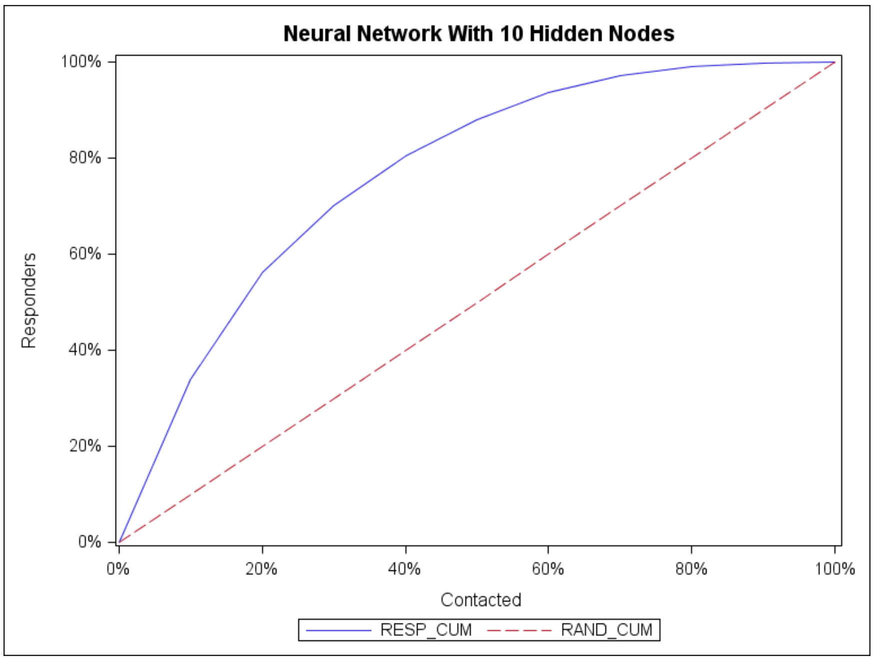


Figure 11. Model gains chart of neural network model with two hidden nodes

For illustrative purpose, the drawing of the actual neural network model with two hidden nodes is depicted in the figure below:

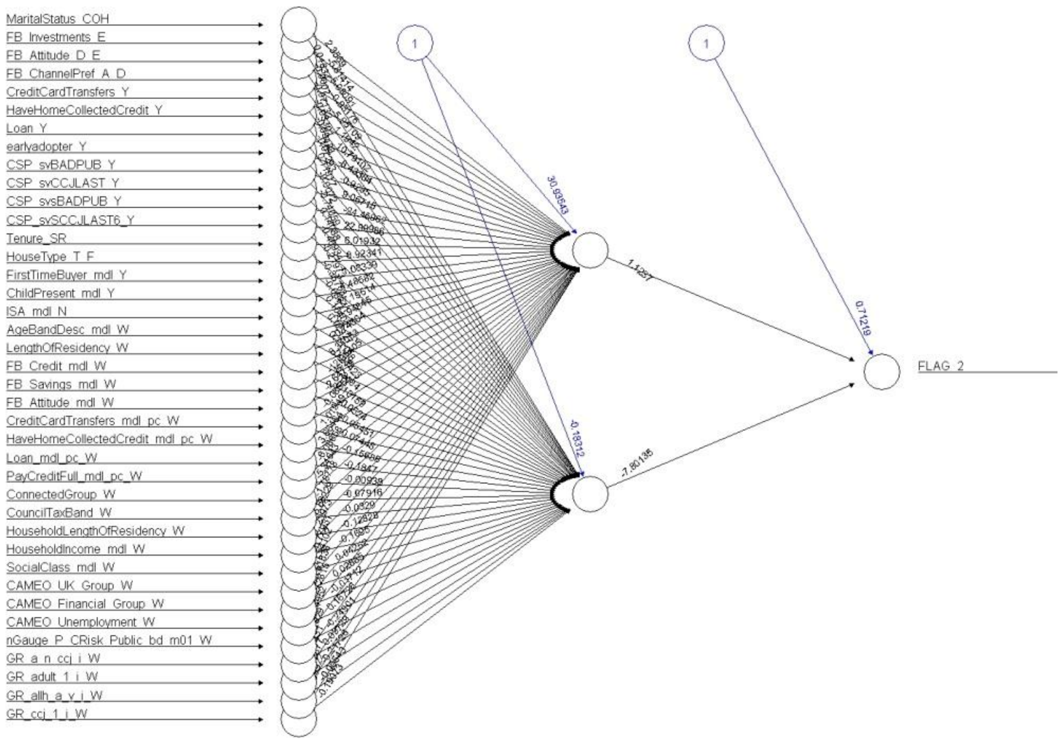


Figure 12. Schematic drawing of neural network with two hidden nodes

3.2. Overall model comparison

Above sections have shown the results of all three models separately and in this section the results are aggregated and compared across the models.

The AUC measure and the gini coefficient of the best models from each of techniques are the following:

Model and Parameters	AUC	Gini	Rank
PCR 1-5 PCs	0,84017	0,68035	3
NN 2 Hidden Nodes	0,86071	0,72142	2
RF 750 Trees	0,86610	0,73220	1

Table 14. Model comparison

In the same way as AUC and gini, model gains charts were compared and plotted on each other:

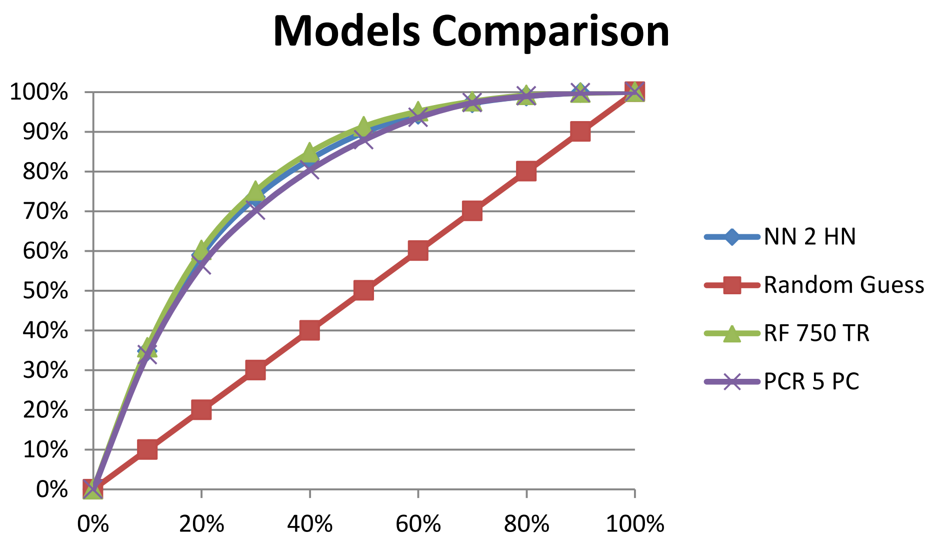


Figure 13. Model comparison in terms of gains chart

Both the table and the figure above are confirming that the best of three compared models is random forest model with 750 trees. Gini coefficient of this model is 0.7322 and appeared to be the best gini coefficient of all of the models that were developed throughout this work. Neural network model with two hidden nodes was performing slightly worse, with gini coefficient of 0.7214, whereas principal component logistic regression model with first five principal components was performing worse than the other two models. Comparison of actual models gain can be seen in the table below:

Decile	Decile Volume	Customers Cumulative % (Logistic Regression)	Customers Cumulative % (Neural Network)	Customers Cumulative % (Random Forests)
1	5472	33,80%	34,7%	35,70%
2	5472	56,40%	58,9%	60,20%
3	5472	70,20%	73,5%	75,10%
4	5472	80,30%	83,0%	84,80%
5	5471	87,90%	89,9%	91,30%
6	5472	93,50%	94,3%	95,10%
7	5472	97,20%	97,2%	97,60%
8	5472	98,90%	98,9%	99,20%
9	5472	99,80%	99,8%	99,70%
10	5471	100,00%	100,0%	100,00%
Total	**54718**	**100,00%**	**100,0%**	**100,00%**

Table 15. Comparison of model gains

From the table above, it can be seen that by selecting record in top three deciles, using different models can result in selecting different percentages of potential customers – using principal component logistic regression, 70.2% of customers are selected, using neural network model, 73.5% of customers are selected and by using the random forest model, 75.1% of potential customers are selected. Using this table, we can confirm once more, that for the given data, random forest model with 750 trees performs the best compare to the other models.

3.3. Comparison of selected records for each model

One of the goals of this work was to find out whether different models are scoring the same records with the highest probabilities, i.e. if the models are selecting the same records as the best prospects. To do that, each record in the validation dataset was assigned a unique reference number and corresponding model decile that was assigned when scoring the validation dataset with all three models. Cross tabulation of each model versus another was performed in terms of deciles and proportion of the same records in +/- 1 deciles was calculated. Below is a table showing a cross-tabulation of deciles of principal component logistic regression and neural network models:

PCLR Decile	Neural Network Decile										
	1	2	3	4	5	6	7	8	9	10	Total
1	4275	1078	45	5	0	0	0	0	0	0	5403
2	989	2533	1306	482	102	20	3	0	0	0	5435
3	103	1099	1908	1466	692	130	45	3	0	0	5446
4	32	418	1253	1591	1367	536	207	53	3	0	5460
5	8	171	631	1139	1487	1160	660	167	31	0	5454
6	1	83	221	538	1071	1402	1353	659	138	3	5469
7	1	27	76	194	522	1206	1608	1291	523	20	5468
8	0	8	12	37	180	790	1214	1852	1248	129	5470
9	0	1	1	8	29	217	364	1298	2344	1208	5470
10	0	0	0	0	8	7	16	146	1183	4111	5471
Total	**5409**	**5418**	**5453**	**5460**	**5458**	**5468**	**5470**	**5469**	**5470**	**5471**	**54546**

Table 16. Cross-tabulation of PCLR and neural network deciles

As it can be seen from the table above, majority of the records are getting either the same decile assigned or at has one decile difference at either side. Similar cross-tabulations were also achieved for neural network versus random forests and principal component logistic regression versus random forests. It can be said from this table that two models are not scoring the same records in opposite manner – i.e. the vast majority of records are assigned either the same or similar decile for both models. The results of these three comparisons can be summarised in the table below which shows the proportion of +/- 1 decile either side for all three models.

	PCR vs. Neural Network	PCR vs. Random Forest	Neural Network vs. Random Forest
Decile	**+/- 1 Decile**		
1	97,3%	97,4%	98,6%
2	86,9%	88,5%	94,3%
3	81,9%	80,4%	85,3%
4	76,8%	72,7%	80,3%
5	71,9%	67,1%	70,1%
6	68,9%	65,4%	67,1%
7	76,3%	69,2%	68,9%
8	81,2%	75,0%	74,4%
9	87,3%	85,8%	80,6%
10	97,2%	94,4%	88,0%
Mean	**82,6%**	**79,6%**	**80,8%**

Table 17. Comparison of deciles across all three models

This shows that for all three models ~80% of the records from the validation data set are getting the same or +/- 1 decile either side.

4. Conclusion

In this work, a problem of creating a customer look-alike model on one of the European credit card companies data

In this work, three different modelling techniques were employed to create a customer look-alike model for binary classification problem – principal component logistic regression, neural network and random forests. To get the best model using each technique, a number of models were create by varying its parameters (combination of principal components in principal component logistic regression, number of hidden nodes in neural network and number of trees in random forests) and the best model was chosen according to area under ROC curve measure, gini coefficient and model gains that are common and widely used measures for evaluation of binary classification.

It turned out that the best models using each of the technique are – principal component logistic regression model with first five principle components, neural network with two hidden nodes and random forests model with 750 trees. These selected best models then were compared in terms of AUC, gini coefficient and gains chart and the result of the comparison showed that random forest model with 750 trees was the top performer among others, neural network with two hidden nodes ranked second and principal component logistic regression with first five principal components turned out to perform worse than the other models.

This result suggests using random forests model for binary classification problem when the data samples are large and the number of variables is high. Also, this result is specific data not only because of high dimensionality but also due to its specificity being real life geodemographical data.

In order to select the best classification model, it is suggested to compare the models using the methodology described in this section rather than using random forest model for any binary classification problem. Because of different data, its size, the variation of variables and the manner of learning algorithms, different models can perform differently given various data samples.

The results of the best models within each technique were also compared in terms of deciles to see if different models are assigning the same or at least similar deciles across the models. The comparison showed that each model was assigning the same or +/- 1 decile each side for

~80% of records in the validation data set. This comparison has shown that different modelling techniques, classifying the records using distinct algorithms, are not contradicting to each other and assigns the same or similar deciles to the majority of records.

5. Further work

It would be interesting to investigate further the records that were assigned the same or similar deciles across different models to see if it is possible to get even better classification results by using the combination of the models.

Also, another modelling technique, called support vector machine (SVM) could be introduced in the comparison as it is also widely used technique for binary classification and it would be interesting to see how it compares to the models that were developed in this work.

Just like bootstrap aggregating was introduced for random forests, it would be interesting to see how this would work on neural network model and if this addition to the model would increase or decrease the overall model performance

References

1. David J. Hand, Robert J. Till. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning*, 45, 171-186, 2001.
2. A.M. Aguilera et al. Using principal component for estimating logistic regression with high-dimensional multicollinear data, *Computational Statistics & Data Analysis* 50 (2006), 1905-1924
3. Swee Lean Chan, Corresponding author at Moonseo Park (2005). Project cost estimation using principal component regression, *Construction Management and Economics*, 23:3, 295-304.
4. Ha, K., Cho, S. and MacLachlan, D. (2005), Response models based on bagging neural networks. J. Interactive Mark., 19: 17–30. doi: 10.1002/dir.20028
5. B. Lariviere, D. Van den Poel. Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications,* 29 (2005) 472-484
6. Escabias, M., Aguilera, A. M. and Valderrama, M. J. (2005), Modeling environmental data by functional principal component logistic regression. Environmetrics, 16: 95–107. doi: 10.1002/env.696
7. R. Genuer, J.M. Poggi, C. Tuleau-Malot. Variable selection using random forests, Pattern Recognition Letters, Vol. 31, Issue 14, October 2015, p. 2225-2236.
8. B. Baesens, S. Viane, D.V. den Poel, J. Vanthienen, G. Dedene. Bayesian neural network learning for repeat purchase modelling in direct marketing, European Journal of Operational Research, Vol. 138, Issue 1, April 2002, p. 191-211.
9. B. S. Everitt, D. C. Howell. Least Square Estimation, Encyclopedia of Statistics in Behavioural Science. John Willey & Sons, Ltd, Chichester, 2005
10. D. C. Lay. Linear Algebra and its Applications, 3rd edition. Pearson Education in South Asia, 2003.
11. L. Breiman. Random forests, *Machine Learning*, 45, 5-32, 2001
12. L. Breiman. Bagging Predictors, *Machine Learning*, 24, 123-140, 1996
13. V. Čekanavičius, G. Murauskas. Statistika ir Jos Taikymai, II d., TEV, Vilnius, 2004.
14. G. Upton, I. Cook. Oxford Dictionary of Statistics, Oxford University Press, New York, 2004.
15. Y. Xie, X, Li, E.W.T Ngai, W. Ying. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36 (2009), 5445-5449.

16. Calle M, Urrea V. Letter to the editor: stability of random forest importance measures. Brief Bioinformatics 2011;12:86–9.
17. Bounds, D., & Ross, D. (1997). Forecasting Customer Response With Neural Networks. Handbook of Neural Computation, G6.2, 1–7.
18. Moutinho, L., Curry, B., Davies, F., & Rita, P. (1994). Neural Network in Marketing. New York: Routledge.
19. Suh, E., Noh, K., & Suh, C. (1999). Customer List Segmentation Using the Combined Response Model. Expert Systems with Applications, 17(2), 89–97.
20. German, S., Bienenstock, E., & Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. Neural Computaion, 4(1), 1–58.

1 Priedas. Kodo fragmentai

Vieno iš atsitiktinių medžių modelių kūrimo R kodas

```
ptm <- proc.time()

rf_750tr_20p=randomForest(as.factor(FLAG_2) ~ ., data=train_data_red_20p,
importance = TRUE, ntree = 750, proximity = FALSE)

rf_750tr_20p_time<-proc.time() - ptm

varImpPlot(rf_750tr_20p_time)

validate_750tr_20p<-predict(rf_750tr_20p,val_data_20p,type="prob")

validate_rf750tr_20p_resp<-cbind(validate_750tr_20p,val_resp_20p)

write.table(validate_rf750tr_20p_resp,file="Results\\rf750tr_20p_R.csv",
sep=',')
```

R kodas vienam iš neuroninių tinkle modeliui kurta

```
ptm <- proc.time()

nnet_10hn<-neuralnet(FLAG_2 ~ MaritalStatus_COH + FB_Investments_E +
FB_Attitude_D_E + FB_ChannelPref_A_D + CreditCardTransfers_Y +
HaveHomeCollectedCredit_Y + Loan_Y + earlyadopter_Y + CSP_svBADPUB_Y +
CSP_svCCJLAST_Y + CSP_svsBADPUB_Y +        CSP_svSCCJLAST6_Y + Tenure_SR +
HouseType_T_F + FirstTimeBuyer_mdl_Y + ChildPresent_mdl_Y + ISA_mdl_N +
AgeBandDesc_mdl_W + LengthOfResidency_W +  FB_Credit_mdl_W +
FB_Savings_mdl_W + FB_Attitude_mdl_W + CreditCardTransfers_mdl_pc_W +
HaveHomeCollectedCredit_mdl_pc_W + Loan_mdl_pc_W + PayCreditFull_mdl_pc_W +
ConnectedGroup_W + CouncilTaxBand_W + HouseholdLengthOfResidency_W +
HouseholdIncome_mdl_W + SocialClass_mdl_W + CAMEO_UK_Group_W +
CAMEO_Financial_Group_W + CAMEO_Unemployment_W +
nGauge_P_CRisk_Public_bd_m01_W + GR_a_n_ccj_i_W + GR_adult_1_i_W +
GR_allh_a_v_i_W + GR_ccj_1_i_W, train_data_red_20p, hidden=10,
threshold=0.05, linear.output=FALSE, stepmax=1e+05)

nnet_10hn_time<-proc.time() - ptm
nnet_10hn_time
print(nnet_10hn)
plot(nnet_10hn)
validate_nnet_10hn<-compute(nnet_10hn,val_data_20p)
validate_nnet_10hn_resp<-cbind(validate_nnet_10hn,val_resp_20p)
ncol(validate_nnet_10hn_resp)
validate_nnet_10hn_resp<-
validate_nnet_10hn_resp[,(ncol(validate_nnet_10hn_resp)-
2):ncol(validate_nnet_10hn_resp)]

names(validate_nnet_10hn_resp)

#confusion matrix
table(round(validate_nnet_10hn_resp$net.result),validate_nnet_10hn_resp$FLA
G_2)

#write.table(validate_nnet_10hn_resp,file="Results\\nnet_10hn_20p_R.csv",
sep=',')
```

R kodas vienam iš pagrindinių komponenčių logistinės regresijos modeliui kurta

```
#creating a tranformation matrix of the centered training dataset
dim(pr_comp)

pr_comp_1_7 = pr_comp[,1:7]

T_1_7<-train_cent%*%pr_comp_1_7

dim(T_1_7)

pca_training_1_7<-cbind(T_1_7,resp_20p_pca)
dim(pca_training_1_7)

summary(pca_training_1_7)

colnames(pca_training_1_7)<-c("pc1", "pc2", "pc3", "pc4", "pc5", "pc6",
"pc7", "resp")

#converting to a data frame
pca_training_1_7<-as.data.frame(pca_training_1_7)

# fitting the logit model now

pca_log_1_7<-glm(resp ~ ., data=pca_training_1_7, family = "binomial")

pca_log_1_7

summary(pca_log_1_7)

res_1_7<-residuals(pca_log_1_7,type="deviance")

plot(predict(pca_log_1_7),res_1_7)

#converting from PC coefficients to original variabbles coefficients

mod_coeff_1_7<-pr_comp_1_7%*%pca_log_1_7$coefficients[2:8]

dim(mod_coeff_1_7)
dim(val_data_20p)

mod_coeff_1_7<-as.matrix(mod_coeff_1_7)

score_1_7<-val_data_20p%*%mod_coeff_1_7

dim(score_1_7)

for (n in 1:54718)
{
score_1_7[n] = score_1_7[n] + pca_log_1_7$coefficients[1]
}

prob_1_7=NULL
for (n in 1:54718)
{
prob_1_7[n] = exp(score_1_7[n])/(1+exp(score_1_7[n]))
}

valid_pca_1_7=cbind(prob_1_7,val_resp_20p)


table(round(prob_1_7),val_resp_20p$FLAG_2)
```

```
write.table(valid_pca_1_7,"C:/Users/pijusk/Documents/Master
Thesis/Results/PCA_Logit_Validation_PC_1-7.csv", sep=",")
```