## Prediction of Data Science Employees’ Salaries using K-Nearest Neighbor, Random Forest and Linear Regression

Sunil Sapkota

*MSC in Data Science*

*Softwarica College*

Dilli bazar, Kathmandu

220440@softwarica.edu.np

***Abstract*—This paper looks at the salaries of the employees who worked in the field of Data Science around the world and try to predict the salaries by analyzing their positions, size of the company, location of the work, employment type and their experience and applying K-Nearest Neighbor, Random Forest and Linear Regression machine learning algorithms. The dataset is taken from Kaggle.com. Data cleaning, encoding feature and target extraction, data visualization and machine learning algorithms were implemented using Python programing language.**

***Keywords—machine learning; python; feature extraction; categorical data; salaries; data science; random forest; k nearest; linear regression.***

1. INTRODUCTION

Data science is one of the most exciting and in-demand fields in the job market. The amount of data generated increased exponentially. Each passing day a huge amount of data is generated. Data is nothing without analysis. The data scientist professional generates, retrieves, stores, and analyses the data. Good analysis of data gives accurate and precious information, which is used to increase the business, predict the future outcomes. Data scientists use various methods and tools to analyze, visualize, and model data to solve real-world problems and generate insights. However, data science is also a highly competitive and dynamic field, where salaries can vary significantly depending on various factors. Therefore, it is important for data scientists to understand the factors that influence their salaries and how to negotiate their income when they get a job.

The number of people pursuing this career has increased in recent decades due to its high demand. Recent advancement in the field of AI like Chat-GPT has increased the curiosity of general people to this field. They want some idea about the salary and relevant of this career for their employment. This paper shows the rate of growth and increment in the salary of professional working in this field of Information Technology.

1. LITERATURE REVIEW

Sayan das, “Salary prediction using regression techniques”,

(2020) has predicted salary of employee using linear and polynomial regression. He found that polynomial regression did a good job for big companies.

TEE Zhen Quan, ”Salary prediction in Data Science field using specialized skills and job benefits – a literature review”,(2022) found that deep learning neural network techniques have shown superiority in processing contextual data and efficient data mining on a larger scale without labelling and structure raw data.

In recent years, there have been several studies on the prediction of salaries using various machine learning algorithms. Data science salaries can range considerably across professionals. According to Glassdoor, the average base pay for data scientists in the U.S. is $117,212 a year. However, the salary can vary from $82K to $167K depending on various factors. K-Nearest Neighbor is a widely used algorithm for regression tasks. A study conducted by Akhtar et al. (2018) used KNN to predict salaries of employees in the IT sector. The study found that KNN performed better than other algorithms, including LR and RF. Similarly, a study conducted by Jiao et al. (2019) used KNN to predict salaries of employees in the finance industry. The study found that KNN outperformed other algorithms, including Decision Tree and Support Vector Machine.

Random Forest is another popular algorithm for regression tasks. A study conducted by Saini et al. (2018) used RF to predict salaries of employees in the IT sector. The study

found that RF outperformed other algorithms, including KNN and LR. Similarly, a study conducted by Li et al. (2018) used RF to predict salaries of employees in the finance industry. The study found that RF outperformed other algorithms, including LR and Gradient Boosting.

Linear Regression is a simple and widely used algorithm for regression tasks. A study conducted by Wang et al. (2020) used LR to predict salaries of employees in the healthcare industry. The study found that LR performed better than other algorithms, including Random Forest and Neural Network

# TABLE 1. DATASET FEATURES

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Description** | **Type** | **Categorical Value Range** |
| 1 | Administrative | Numeric |  |
| 2 | Administrative Duration | Numeric |  |
| 3 | Informational | Numeric |  |
| 4 | Informational Duration | Numeric |  |
| 5 | Product Related | Numeric |  |
| 6 | Product Related Duration | Numeric |  |
| 7 | Bounce Rates | Numeric |  |
| 8 | Exit Rates | Numeric |  |
| 9 | Page Value | Numeric |  |
| 10 | Special Day | Numeric |  |
| 11 | Month | Categorical | February to  December |
| 12 | Operating Systems | Categorical | 1 to 8 |
| 13 | Browser | Categorical | 1 to 13 |
| 14 | Region | Categorical | 1 to 9 |
| 15 | Traffic Type | Categorical | 1 to 20 |
| 16 | Visitor Type | Categorical | New,  Returning, Other |
| 17 | Weekend | Categorical | True, False |
| 18 | Revenue | Categorical | True, False |

III. DATA PREPARATION

This section covers data analysis on class imbalance and overcoming imbalance issues, dealing with categorical data and categorical data encoding, analysis of numerical values and data scaling, feature analysis, and feature extraction.

# A. Class Imbalance

Haibo He in the book Self-adaptive Systems for Machine Intelligence (2011) explains that class imbalance caused machine learning algorithms to underperform and this issue must be addressed before proceeding to implement algorithms on the dataset. Haibo He suggests to use random under-sampling, random over-sampling or SMOTE oversampling techniques. In this paper, class imbalance analysis was carried out on the dataset usingpython*.* Example code was obtained from Kaggle written by Manish Kumar (2018) and modified to suit dataset. The analysis identified 10422 visits that lead to false revenue (non-purchase) and just 1908 that led to true revenue (Purchase). Results visualized in *Fig 1* show class imbalance on the dataset. Code for importing data set and plotting class imbalance see appendix IX.A.2)

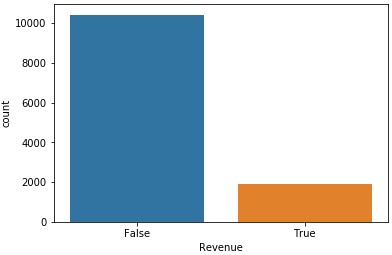


Fig 1. Class Imbalance

To overcome these three techniques were used as suggested by He. For source code see appendix IX.A.7) *1) Random Under-sampling*

Random under-sampling is performed by randomly selecting a number of samples from the majority class to match the number of samples of a minority class. In this paper 1908 instances were selected where revenue was false and merged with 1908 instances where revenue is True, to create a new dataset of 3816 instances. *2) Random Over-sampling*

Random over-sampling is performed by randomly duplicating instances of minority class to match the number of instances from majority class. In this paper, 10422 instances were selected at random with replacement from minority class and merged with 10422 instances of majority class to create a new dataset containing 20844 instances. *3) SMOTE Over-sampling*

SMOTE over-sampling algorithm creates synthetic instances by a true revenue example and selecting the nearest neighbor that is also a True revenue example and creating an instance between those two records. A visual example of SMOTE displayed in *Fig 2*.

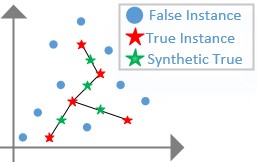


Fig 2. SMOTE Example

In this paper, SMOTE was implemented using the algorithm from python imbalanced learning library. Synthetic instances of True revenue class were created to match the number of true revenue class with false revenue class and new dataset produced with 20844 instances.

# B. Categorical Feature Encoding

Many machine learning algorithms only work with numeric values, but categorical data can be represented as text. Even if categorical data is represented as a number (i.e. months 1-12) machine learning algorithm could give higher weight to December as it carries number 12 than January 1 and not classify them in the same way. To avoid this categorical data must be encoded into binary values and to simplify further dummy variable can be removed in order to reduce dimensionality of the dataset (Brownlee 2017, Alpaydın 2014). In this paper, eight categorical features have been encoded using python library scikit-learn utilizing LabelEncoder to encode labels into numbers (e.g. month from text February to number 2 and Tree to 1) and OneHotEncoder to convert values to binary as per example displayed in *TABLE*

*2*.

## TABLE 2 ENCODING CATEGORICAL FEATURES TO BINARY

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Text** | → | **Numerical** | → | **Binary** | | |
| ***Month*** | ***Month*** | ***February*** | ***March*** | ***April*** |
| February | 2 | 1 | 0 | 0 |
| March | 3 | 0 | 0 | 1 |
| April | 4 | 0 | 1 | 0 |

Once encoding is completed the dummy variables are removed. (e. g. by removing one month it is not lost, it is represented by zeros in remaining month columns) the example displayed in *TABLE 3* show April being removed, but it can be identified by zero values in February and March columns same technique was applied in online shoppers dataset. For source code on encoding and dummy variable removal see appendix IX.A.5)

## TABLE 3 DUMMY VARIABLE REMOVAL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **February** | **March** | **April** | → | **February** | **March** |
| 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |

# C. Data Standardisation

In order to evaluate each feature equally, data standardization must be performed to ensure all the features are on the same scale and none of the features are given higher weighting by the algorithms. This could happen as 100 is not the same as 1 and in the formula 100 would have a higher weighting for prediction. Standardization is done by subtracting the mean from each feature and divide by the standard deviation (1) (Hackeling 2014).

𝑥′ = 𝑥−𝑥̅ (1)

𝜎

In this paper, standardization was completed utilizing python Standard Scaler class from preprocessing module in scikit-learn library.

# D. Dimensionality Reduction and Feature Extraction

Dimensionality reduction is performed to improve the speed of data processing by reducing the amount of data to be processed while minimizing information loss, it also enables users to reduce data to two dimensions or three dimensions for visualization purposes. Dimensionality reduction can be done using Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA) or Kernel PCA (Gnanadesikan 1988, Hackeling 2014). For source code of techniques see appendix IX.A.7) *1) LDA*

LDA is a supervised learning technique that analyses and identifies the features that have the highest class separation. In python, LDA was implemented using scikit-learn repository. LDA analysis was run however it has identified that features are collinear meaning features are closely correlated and it is unable to separate them, therefore, LDA is not an option for extracting features from the online shopper’s data set. A paper by Naes and Mevik (2001) states that this issue can be overcome by applying PCA. *2) PCA*

PCA analyses features identifies their variance and sorts by highest variance. In python, PCA was implemented using scikit-learn repository and results displayed in *Fig. 3* and *Fig. 4* obtained.

Fig. 3 PCA Individual Component Variance

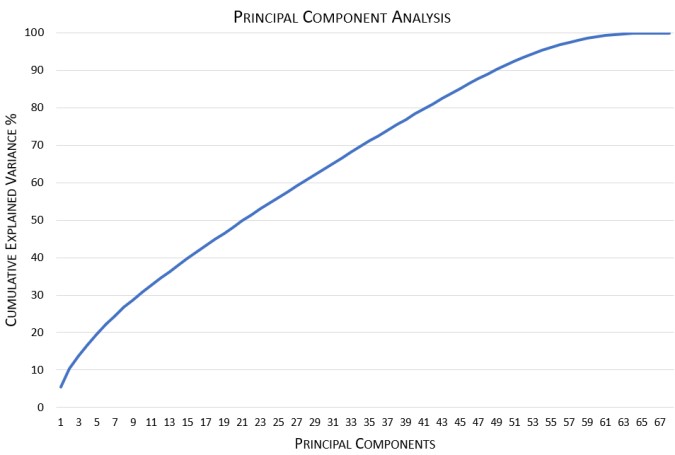


Fig. 4 PCA Cumulative Variance

Results show that most of the variables carry similar variance, meaning variables are almost equally important and only minimal dimensionality reduction can be performed. *3) Kernel PCA*

Kernel PCA extends PCA by incorporating a kernel allowing to separate features that are non-linearly separable.

In python Kernel PCA was implemented using scikit-learn repository and results displayed in *Fig. 5* and *Fig. 6* (Hill and Lewicki 2006).

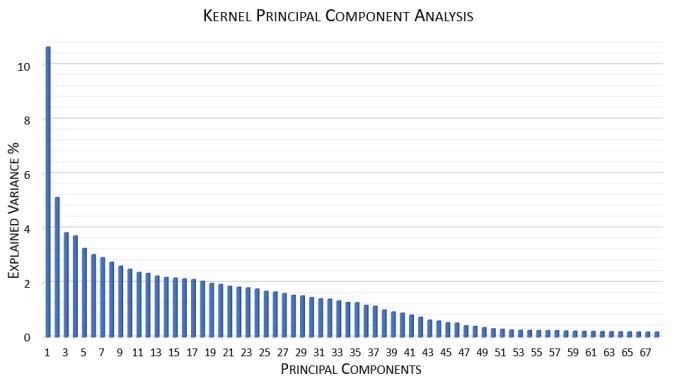
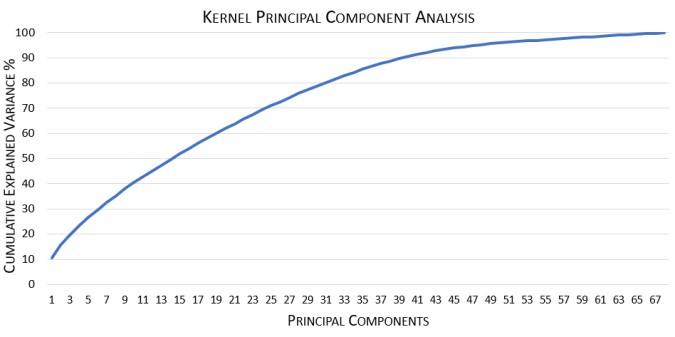
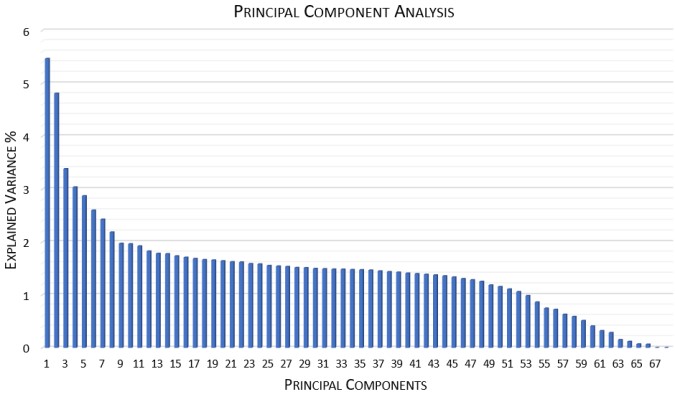


Fig. 5 Kernel PCA Individual Component Variance

Fig. 6 Kernel PCA Cumulative Variance

Analyzing results and comparing to PCA differences can be identified. Kernel PCA displays 38 components with a variance above 1% where PCA had 52 and Kernel PCA show that 40 components add up 90% of cumulative variance wherein PCA this number is 49. The comparison suggests that Kernel PCA performs better and it is the technique to be used for online shopper’s dataset, therefore LDA and PCA will not be used in this paper. Kernel PCA with 38 features will be used as it reduces dataset by 30 features while still having 88.77% cumulative variance.



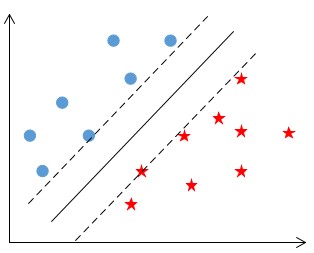
IV. MACHINE LEARNING CLASSIFICATION TECHNIQUES

# A. K-Nearest Neighbor (K-NN)

K-NN algorithm compares given instance to the *k* number of nearesttraining instances and classifies the new instance based on majority voting of the nearest neighbor’s classes or by distance weighting. Distances are calculated using the Euclidean distance metric. A number of neighbors must be specified, and most commonly used number is 5 (Hill and Lewicki 2006).

# B. Support Vector Machines (SVM)

SVM is used to classify instances by linearly separating them with the highest margin possible between the class instances. When data is non-linear separable SVM uses a kernel and introduces another dimension in order to make the data separable with a hyperplane this type of action is referred to as kernel trick. Examples of linear separation and nonlinear separation utilizing kernel displayed in Fig 7 *Fig. 8*. Figures are for example purposes and do not represent online shoppers dataset.



*Fig 7 SVM Linear Separation*

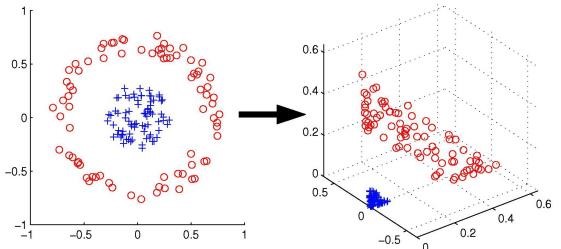


Fig. 8 SVM Non-linear Separation Using Kernel

# C. Ensembles 1) Random Forest

Random forest is built on decision trees, decision three starts classification with the root variable and through binary decisions adds branches (variables) until it arrives at the leaf (class). Random Forest works by training multiple decision tree models, combining their classification results and using majority voting to arrive at final prediction (Gra̧bczewski 2014).

# 2) Adaptive Boosting (ADA Boost)

ADA Boost most popular boosting algorithm. It uses other learning algorithms by repeatedly training them multiple times and each time focusing on misclassified data and adjusting to improve classification turning the weak learning algorithms into strong (Gra̧bczewski 2014). For online shopper’s classification ,ADA Boost was used with Random Forest classifier.

# 3) Extremely Randomised Trees

The algorithm is built on decision trees. It stands out from other ensemble tree-based models such as Random Forest and ADA Boost by splitting tree nodes at random and uses whole training data set rather than bootstrap version (Geurts et al. 2006).

V. APPLICATION OF TECHNIQUES AND RESULTS Algorithms were implemented, and results obtained using HP Omen computer (laptop) equipped with Intel Core i76700HQ CPU with 2.60 GHz and 16GB RAM.

Models were trained using parameters that deliver the best results, parameters were obtained using grid search. Confusion Matrices, ROC curve (except SVM) and model metrics for all models were obtained and are presented for each model.

All models were trained using 38 best features and each model was trained three times to evaluate which class imbalance technique is best and to obtain the best results.

Model evaluation metrics:

* 10-fold cross-validation was performed and model accuracy obtained.
* Model Accuracy
* Model Precision
* Specificity
* Sensitivity / Recall
* F1 Measure
* Geometric Mean Measure
* Mathews Correlation Coefficient
* Time taken to train and test. This includes 10 fold cross-validation and metrics calculations

# A. Support Vector machines

Using python libraries SVC classifier, support vector machine model was trained and results presented in *Fig. 9* and *TABLE 4*.

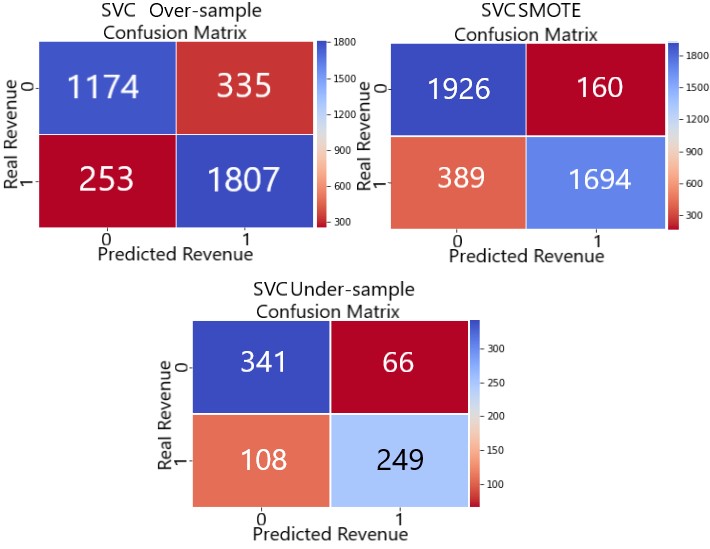


Fig. 9 Confusion Matrices SVM Classifier

TABLE 4 SUPPORT VECTOR MACHINES RESULTS SUMMARY

|  |  |  |  |
| --- | --- | --- | --- |
| **S** | **upport Vector Machines** | |  |
|  | ***Algorithm Parameters*** | |  |
| ***C*** | 4 | 10 | 4 |
| ***Kernel*** | Gaussian | Gaussian | Gaussian |
| ***Gamma*** | 0.3 | 1 | 1 |
|  | ***Results*** | |  |
| ***Class Imbalance*** | SMOTE | Over-sample | Undersample |
| ***No of Test Examples*** | 4169 | 4169 | 764 |
| ***True Positive*** | 1694 | 1807 | 249 |
| ***True Negative*** | 1926 | 1774 | 341 |
| ***False Positive*** | 160 | 335 | 66 |
| ***False Negative*** | 389 | 253 | 108 |
| ***Correctly Classified*** | 3620 | 3581 | 590 |
| ***Miss Classified*** | 549 | 588 | 174 |
|  | ***Cross-Validation 10-Fold*** | |  |
| ***Accuracy*** | 0.86 | 0.85 | 0.76 |
| ***Variation +/-*** | 0.08 | 0.1 | 0.12 |
|  | ***Metrics*** | |  |
| ***Accuracy*** | 0.8683 | 0.8590 | 0.7723 |
| ***Precision*** | 0.9137 | 0.8436 | 0.7905 |
| ***Specificity*** | 0.9233 | 0.8412 | 0.8378 |
| ***Sensitivity / Recall*** | 0.8133 | 0.8772 | 0.6975 |
| ***F1 Measure*** | 0.8606 | 0.8601 | 0.7411 |
| ***G Measure*** | 0.8665 | 0.8590 | 0.7644 |
| ***Matthews Corr Coef*** | 0.7411 | 0.7186 | 0.5426 |
| ***Time taken*** | 136.25 | 169.78 | 5.40 |

Results show that SVM performed best using SMOTE over-sampling technique. The random over-sampling technique had very similar results and random undersampling produced the worst results. Undersampling was expected to produce the worst results as the training set is significantly smaller when compared to Random oversampling and SMOTE over-sampling.

Results also indicate that the dataset is challenging and basic models such as support vector machines have difficulties in learning and delivering perfect predictions. For source code on running the techniques refer to appendices IX.A.4) for modeling function and producing results, IX.A.8) for source code using SMOTE, IX.A.9) for source code using over-sampling, IX.A.10) for source code using undersampling, IX.A.11) for grid search.

# B. K-Nearest Neighbor

Using python libraries K-Nearest Neighbor classifier was trained and results presented in *Fig. 10*, *Fig. 11* and *TABLE 5*

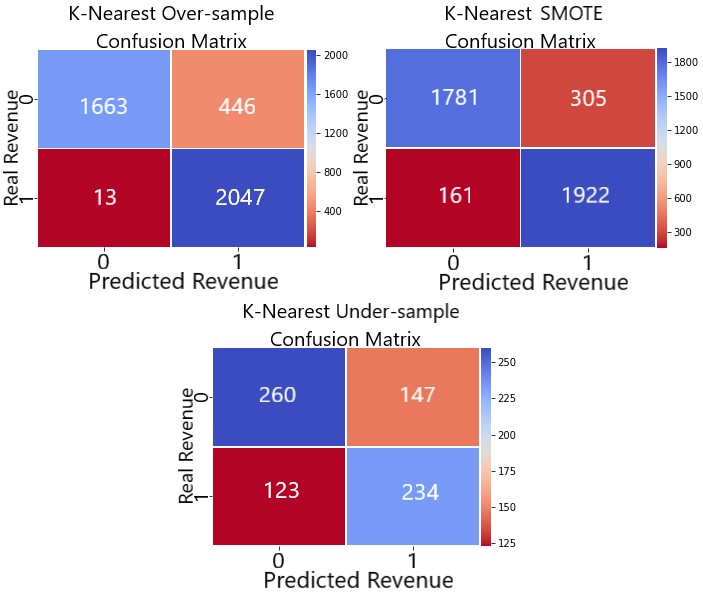


Fig. 10 Confusion Matrices K-Nearest Classifier

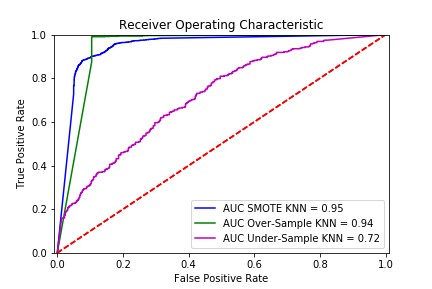


Fig. 11 ROC Curve K-Nearest

## TABLE 5 K-NEAREST RESULTS SUMMARY

|  |  |  |  |
| --- | --- | --- | --- |
|  | **K-Nearest Neighbor** | |  |
|  | **Algorithm Parameters** | |  |
| ***Neighbors*** | 3 | 3 | 5 |
| ***Weights*** | Distance | Distance | Distance |
| ***Distance*** | Euclidean | Euclidean | Euclidean |
|  | ***Results*** | |  |
| ***Class Imbalance*** | SMOTE | Oversampling | Under-  Sampling |
| ***No of Test Examples*** | 4169 | 4169 | 764 |
| ***True Positive*** | 1922 | 2047 | 234 |
| ***True Negative*** | 1781 | 1663 | 260 |
| ***False Positive*** | 305 | 446 | 147 |
| ***False Negative*** | 161 | 13 | 123 |
| ***Correctly Classified*** | 3703 | 3710 | 494 |
| ***Miss Classified*** | 466 | 459 | 270 |
| ***Area Under ROC*** | 0.9460 | 0.9384 | 0.7181 |
|  | ***Cross-Validation 10-Fold*** | |  |
| ***Accuracy*** | 0.87 | 0.88 | 0.60 |
| ***Variation +/-*** | 0.07 | 0.06 | 0.18 |
|  | ***Metrics*** | |  |
| ***Accuracy*** | 0.8882 | 0.8899 | 0.6466 |
| ***Precision*** | 0.8630 | 0.8211 | 0.6142 |
| ***Specificity*** | 0.8538 | 0.7885 | 0.6388 |
| ***Sensitivity / Recall*** | 0.9227 | 0.9937 | 0.6555 |
| ***F1 Measure*** | 0.8919 | 0.8992 | 0.6341 |
| ***G Measure*** | 0.8876 | 0.8852 | 0.6471 |
| ***Matthews Corr Coef*** | 0.7783 | 0.7976 | 0.2937 |
| ***Time taken*** | 15.0816 | 7.7202 | 1.0166 |

K-nearest model results were very similar in comparison to SVM. In this scenario, random over-sampling technique delivered the best results however it still has misclassified 12% of the test cases.

# C. Random Forest

Random forest algorithm from the scikit-learn repository in python was trained, tested and results obtained. Confusion matrices presented in *Fig. 12*, ROC curve in *Fig. 13* and results summary presented in *TABLE 6*.

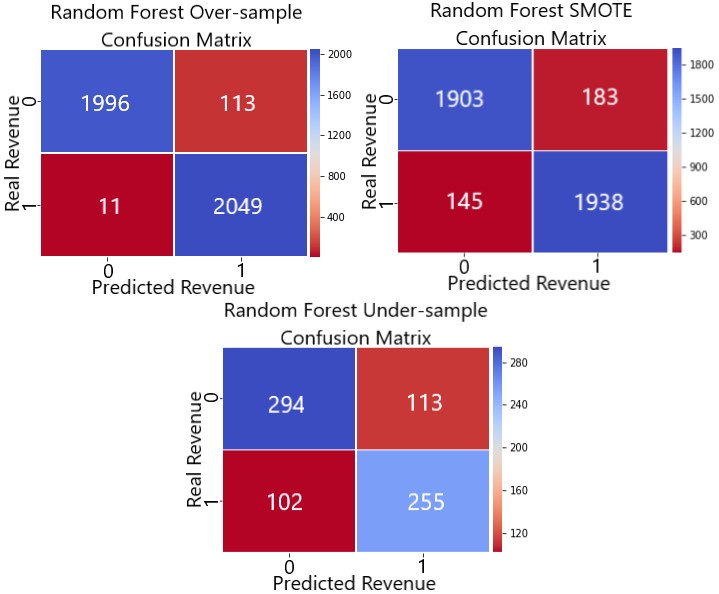


Fig. 12Confusion Matrices Random Forest

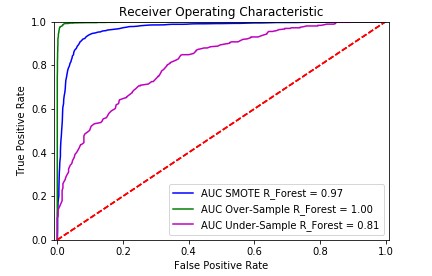


Fig. 13 ROC Curve Random Forest

## TABLE 6 RANDOM FOREST RESULTS

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Random Forest** | |  |
|  | **Algorithm Parameters** | |  |
| ***No. Estimators*** | 145 | 123 | 110 |
|  | ***Results*** | |  |
| ***Class Imbalance*** | SMOTE | Oversampling | Under-sampling |
| ***No of Test Examples*** | 4169 | 4169 | 764 |
| ***True Positive*** | 1938 | 2049 | 255 |
| ***True Negative*** | 1903 | 1996 | 294 |
| ***False Positive*** | 183 | 113 | 113 |
| ***False Negative*** | 145 | 11 | 102 |
| ***Correctly Classified*** | 3841 | 4045 | 549 |
| ***Miss Classified*** | 328 | 124 | 215 |
| ***Area Under ROC*** | 0.9666 | 0.9983 | 0.8138 |
|  | ***Cross-Validation 10-Fold*** | |  |
| ***Accuracy*** | 0.90 | 0.97 | 0.69 |
| ***Variation +/-*** | 0.07 | 0.04 | 0.19 |
|  | ***Metrics*** | |  |
| ***Accuracy*** | 0.9213 | 0.9703 | 0.7186 |
| ***Precision*** | 0.9137 | 0.9477 | 0.6929 |
| ***Specificity*** | 0.9123 | 0.9464 | 0.7224 |
| ***Sensitivity / Recall*** | 0.9304 | 0.9947 | 0.7143 |
| ***F1 Measure*** | 0.9220 | 0.9706 | 0.7034 |
| ***G Measure*** | 0.9213 | 0.9702 | 0.7183 |
| ***Matthews Corr Coef*** | 0.8428 | 0.9417 | 0.4360 |
| ***Time taken*** | 117.74 | 103.56 | 20.76 |

Comparing Random Forest results with KNN and SVM there is a significant improvement in the results. Comparing different class imbalance techniques of Random Forest, oversampling outperformed other two in almost all the areas.

# D. Adaptive Boosting

The adaptive boosting model was based on already well performing random forest algorithm. Adaptive boosting was used with SAMME parameter algorithm which focuses on misclassified instances and SAMME.R on their probabilities. SAMME.R is faster to run, but grid search identified that SAMME delivers better results (Scikit-Learn 2018). The model was trained, tested and results obtained. Confusion matrices displayed in *Fig. 14*, ROC curve *Fig. 15* and results summary in *TABLE 7*.

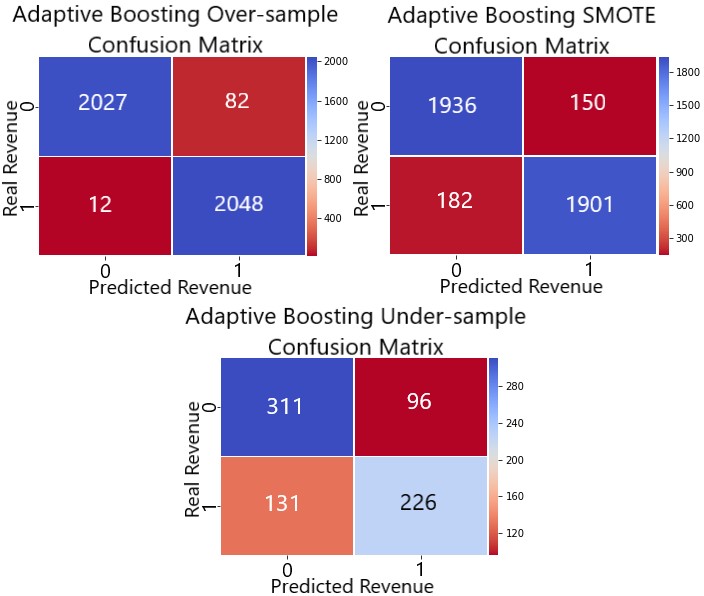


Fig. 14 Confusion Matrices Adaptive Boosting

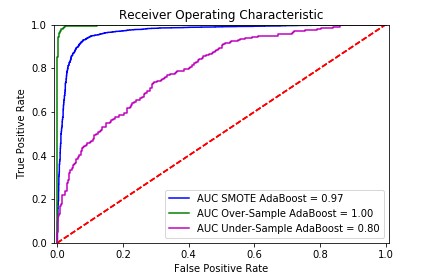


Fig. 15 ROC Curve Adaptive Boosting

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Adaptive Boosting** | | |
|  | **Algorithm Parameters** | | |
| ***Base Estimator*** | Random Forest | | |
| ***No. Estimators*** | 123 | 123 | 84 |
| ***Algorithm*** | SAMME | | |
|  | ***Results*** | | |
| ***Class Imbalance*** | SMOTE | Oversampling | Under-sampling |
| ***No of Test Examples*** | 4169 | 4169 | 764 |
| ***True Positive*** | 1901 | 2048 | 226 |
| ***True Negative*** | 1936 | 2027 | 311 |
| ***False Positive*** | 150 | 82 | 96 |
| ***False Negative*** | 182 | 12 | 131 |
| ***Correctly Classified*** | 3837 | 4075 | 537 |
| ***Miss Classified*** | 332 | 94 | 227 |
| ***Area Under ROC*** | 0.9675 | 0.9980 | 0.7989 |
|  | ***Cross-Validation 10-Fold*** | | |
| ***Accuracy*** | 0.90 | 0.98 | 0.79 |
| ***Variation +/-*** | 0.08 | 0.03 | 0.19 |
|  | ***Metrics*** | | |
| ***Accuracy*** | 0.9204 | 0.9775 | 0.7029 |
| ***Precision*** | 0.9269 | 0.9615 | 0.7019 |
| ***Specificity*** | 0.9281 | 0.9611 | 0.7641 |
| ***Sensitivity / Recall*** | 0.9126 | 0.9942 | 0.6331 |
| ***F1 Measure*** | 0.9197 | 0.9776 | 0.6657 |
| ***G Measure*** | 0.9203 | 0.9775 | 0.6955 |
| ***Matthews Corr Coef*** | 0.8408 | 0.9555 | 0.4013 |
| ***Time taken*** | 203.37 | 176.19 | 16.96 |

TABLE 7 ADAPTIVE BOOSTING RESULTS

As expected, results adaptive boosting have delivered improved from results Random Forest results. When comparing different class imbalance techniques random over-sample outperformed other techniques in all measures.

# E. Extremely Randomized Trees

Extremely randomized trees also part of the ensembles group of algorithms. The model was trained, tested and results obtained. Confusion matrices displayed in *Fig. 16*, ROC curve in *Fig 17* and results summary in *TABLE 8*

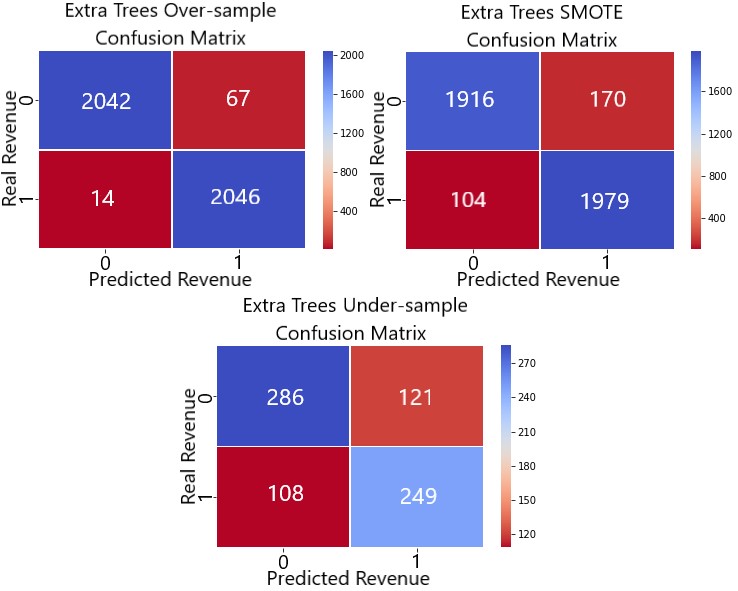


Fig. 16 Extremely Randomized Trees

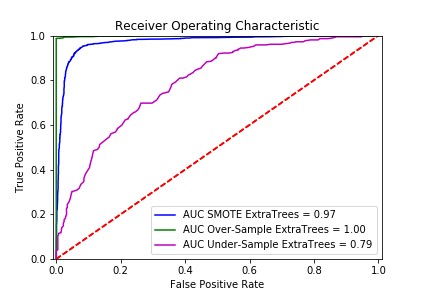


Fig 17 ROC Curve Extremely Randomized Trees

## TABLE 8 EXTREMELY RANDOMIZED TREES RESULTS

|  |  |  |  |
| --- | --- | --- | --- |
| **Extremely Randomized Trees** | | |  |
| **Algorithm Parameters** | | |  |
| ***No. Estimators*** | 2000 | 144 | 162 |
| ***Results*** | | |  |
| ***Class Imbalance*** | SMOTE | Oversampling | Under-sampling |
| ***No of Test Examples*** | 4169 | 4169 | 764 |
| ***True Positive*** | 1979 | 2046 | 249 |
| ***True Negative*** | 1916 | 2042 | 286 |
| ***False Positive*** | 170 | 67 | 121 |
| ***False Negative*** | 104 | 14 | 108 |
| ***Correctly Classified*** | 3895 | 4088 | 535 |
| ***Miss Classified*** | 274 | 81 | 229 |
| ***Area Under ROC*** | 0.9721 | 0.9989 | 0.7910 |
| ***Cross-Validation 10-Fold*** | | |  |
| ***Accuracy*** | 0.92 | 0.98 | 0.66 |
| ***Variation +/-*** | 0.07 | 0.02 | 0.2 |
| ***Metrics*** | | |  |
| ***Accuracy*** | 0.9343 | 0.9806 | 0.7003 |
| ***Precision*** | 0.9209 | 0.9683 | 0.6730 |
| ***Specificity*** | 0.9185 | 0.9682 | 0.7027 |
| ***Sensitivity / Recall*** | 0.9501 | 0.9932 | 0.6975 |
| ***F1 Measure*** | 0.9353 | 0.9806 | 0.6850 |
| ***G Measure*** | 0.9342 | 0.9806 | 0.7001 |
| ***Matthews Corr Coef*** | 0.8690 | 0.9615 | 0.3995 |
| ***Time taken*** | 532.67 | 32.14 | 8.34 |

Extremely randomized trees delivered excellent results. Its ROC curve is almost vertical, visibly different from other models. The random over-sampling technique produced excellent results outperforming SMOTE and under-sampling in all areas.

VI. CONCLUSION

Random over-sampling class imbalance resolution technique performed best across all the models (except SVM). Results for random over-sampling has been combined and presented in *TABLE 9* for comparison.

## TABLE 9 RANDOM-OVERSAMPLING RESULTS SUMMARY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **SVM** | **KNN** | **Random**  **Forest** | **Extra Trees** | **Adaptive Boosting** |
| ***No of Test Examples*** | 4169 | 4169 | 4169 | 4169 | 4169 |
| ***True Positive*** | 1807 | 2047 | 2049 | 2046 | 2048 |
| ***True Negative*** | 1774 | 1663 | 1996 | 2042 | 2027 |
| ***False Positive*** | 335 | 446 | 113 | 67 | 82 |
| ***False Negative*** | 253 | 13 | 11 | 14 | 12 |
| ***Correctly Classified*** | 3581 | 3710 | 4045 | 4088 | 4075 |
| ***Miss Classified*** | 588 | 459 | 124 | 81 | 94 |
| ***Area Under ROC*** | N/A | 0.9384 | 0.9983 | 0.9989 | 0.9980 |
| ***Cross-Validation 10-Fold*** | | | | | |
| ***Accuracy*** | 0.85 | 0.88 | 0.97 | 0.98 | 0.98 |
| ***Variation +/-*** | 0.10 | 0.06 | 0.04 | 0.02 | 0.03 |
| ***Metrics*** | | | | | |
| ***Accuracy*** | 0.8590 | 0.8899 | 0.9703 | 0.9806 | 0.9775 |
| ***Precision*** | 0.8436 | 0.8211 | 0.9477 | 0.9683 | 0.9615 |
| ***Specificity*** | 0.8412 | 0.7885 | 0.9464 | 0.9682 | 0.9611 |
| ***Sensitivity / Recall*** | 0.8772 | 0.9937 | 0.9947 | 0.9932 | 0.9942 |
| ***F1 Measure*** | 0.8601 | 0.8992 | 0.9706 | 0.9806 | 0.9776 |
| ***G Measure*** | 0.8590 | 0.8852 | 0.9702 | 0.9806 | 0.9775 |
| ***Matthews Corr Coef*** | 0.7186 | 0.7976 | 0.9417 | 0.9615 | 0.9555 |
| ***Time taken (seconds)*** | 169.78 | 7.72 | 103.56 | 32.14 | 176.19 |

Review of random over-sampling results show that ensemble models produced very similar results and extremely randomized model had the best results in most areas, especially in the time taken to process. Extremely randomized trees were more than three times faster when compared to random forest and five times faster when compared to adaptive boosting.

Based on results obtained the best model to use for online shoppers’ intentions prediction is Extremely Randomized Trees with a dataset that had its classes balanced using random over-sampling technique and with just 38 features that were extracted using Kernel PCA.

1. FUTURE RESEARCH

The trained model already performed well, however deep learning methods or neural networks could be applied to it in attempt to get better performance as well as more in-depth grid search technique combined with piping which allows to search for an optimum number of components for an algorithm for each of the specific parameters.

1. REFERENCES

Alpaydın, E. (2014) *Introduction to Machine Learning*. 3. ed. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press

Brownlee, J. (2017) ‘Why One-Hot Encode Data in Machine Learning?’ [27 July 2017] available from <https://machinelearningmastery.com/why-onehot-encode-data-in-machine-learning/> [13 December 2018]

Geurts, P., Ernst, D., and Wehenkel, L. (2006) ‘Extremely

Randomized Trees’. *Machine Learning* 63 (1), 3–42

Gnanadesikan, R. (1988) *Discriminant Analysis and Clustering.* [online] National Academies Press. available from

<http://public.eblib.com/choice/publicfullrecord.as px?p=4388301> [7 December 2018]

Gra̧bczewski, K. (2014) *Meta-Learning in Decision Tree Induction*. Studies in computational intelligence Volume 498. Cham ; New York: Springer

# Hackeling, G. (2014) Mastering Machine Learning with Scikit-Learn: Apply Effective Learning Algorithms to Real-World Problems Using Scikit-Learn. Packt open source. Birmingham: Packt Publ

He, H. (2011) *Self-Adaptive Systems for Machine Intelligence* [online] Hoboken, N.J.: Wiley. available from <http://site.ebrary.com/id/10484685> [7 December 2018]

# Hill, T. and Lewicki, P. (2006) Statistics: Methods and Applications: A Comprehensive Reference for Science, Industry, and Data Mining. Tulsa, OK: StatSoft

Kanter, J. (2018) *Cambridge Analytica Bosses Were Secretly Filmed Boasting about How They Helped Trump Win the US Election* [online] available from <https://www.businessinsider.com/cambridgeanalytica-boasts-won-trump-election-facebookdata-2018-3> [13 December 2018]

Kumar, M. (2018) *How To Handle Imbalance Data : Study in Detail | Kaggle* [online] available from <https://www.kaggle.com/gargmanish/how-tohandle-imbalance-data-study-in-detail> [13 December 2018]

Naes, T. and Mevik, B.-H. (2001) ‘Understanding the Collinearity Problem in Regression and Discriminant Analysis: COLLINEARITY

PROBLEM IN REGRESSION AND

DISCRIMINANT ANALYSIS’. *Journal of Chemometrics* 15 (4), 413–426

# Scikit-Learn (2018) sklearn.ensemble.AdaBoostClassifier — Scikit-Learn 0.20.1 Documentation [online]

available from <https://scikit-

learn.org/stable/modules/generated/sklearn.ensemb le.AdaBoostClassifier.html> [13 December 2018]

# Statista (2018) Global Retail E-Commerce Market Size 2014-

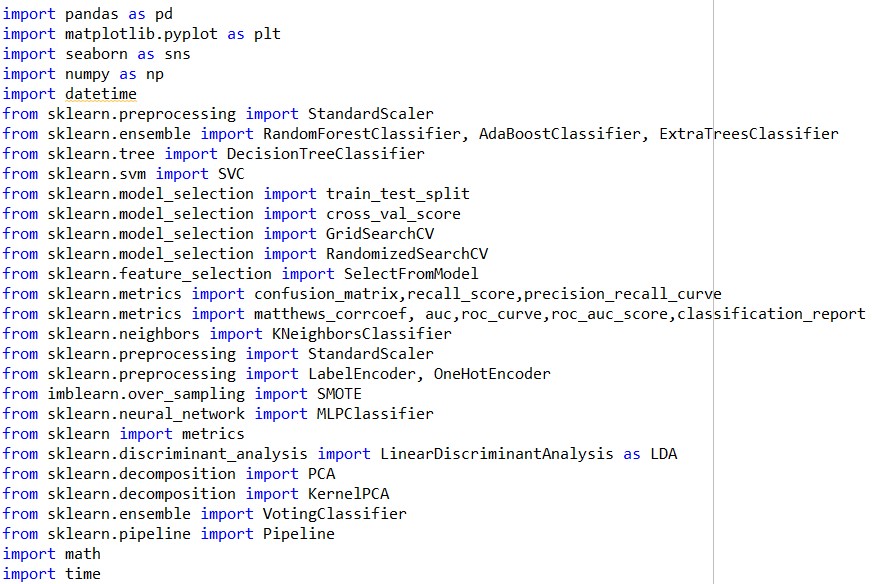
*2021* [online] available from

<https://www.statista.com/statistics/379046/world wide-retail-e-commerce-sales/> [13 December

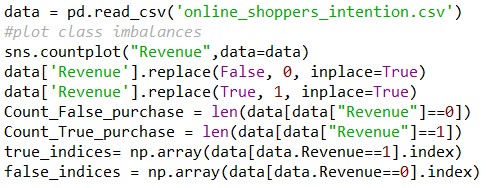
2018]

IX. APPENDIX

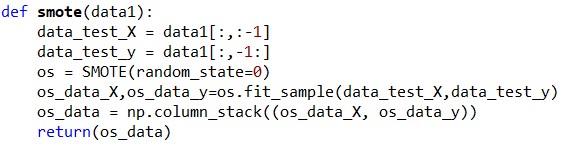
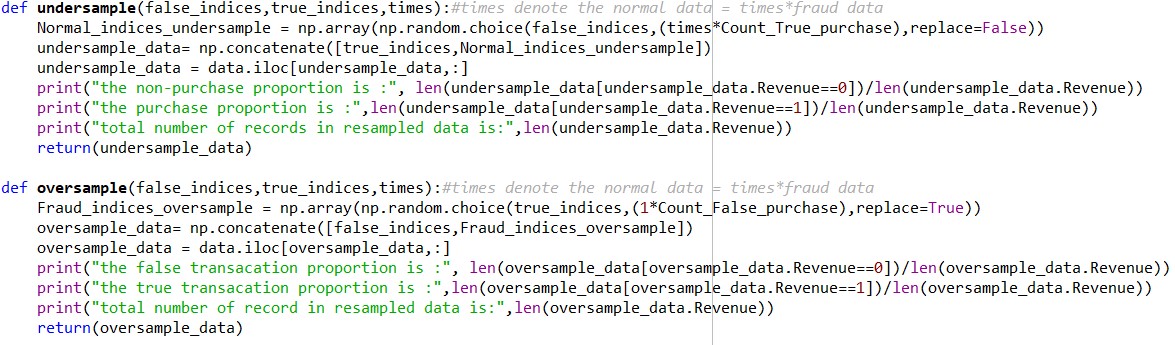
# A. Source code 1) Libraries that were imported as part of the experiment



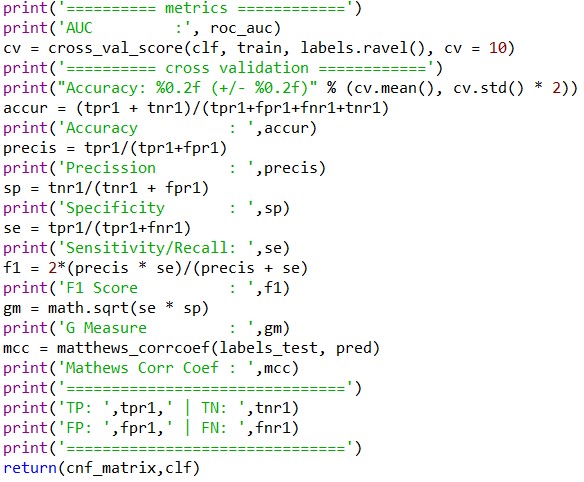
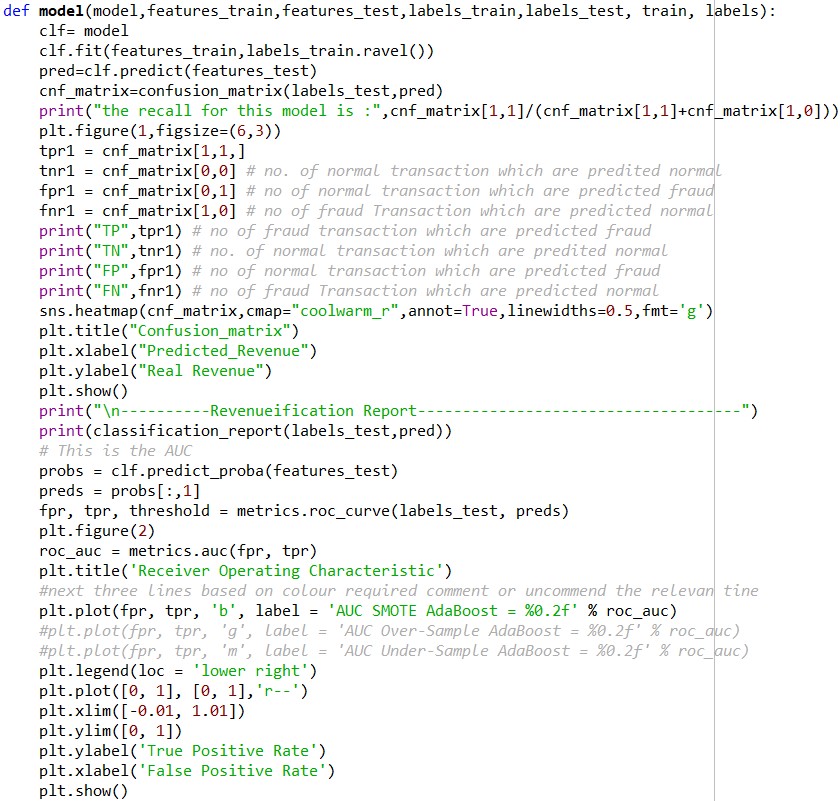
# 2) Importing Dataset, Pot Class Imbalance and Prepare for Over/Under-sampling

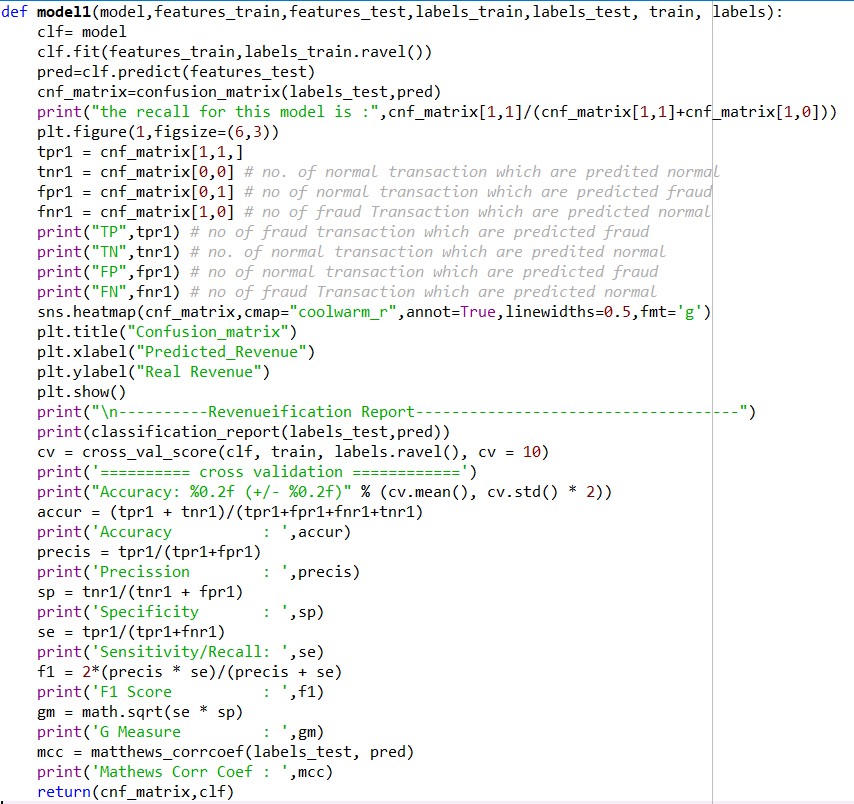


# 3) Functions Defined and Used for Over/Under-Sampling

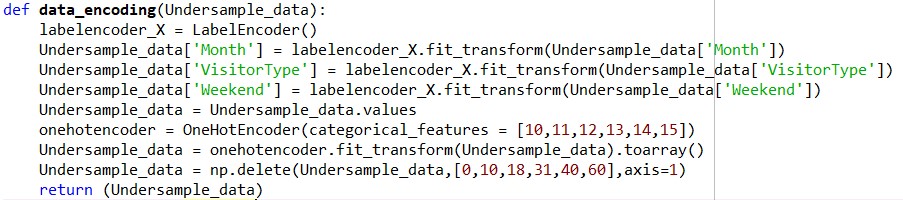


*4) Model and Model1 Functions Used for Training, Testing, Validation, and Metrics Calculation*

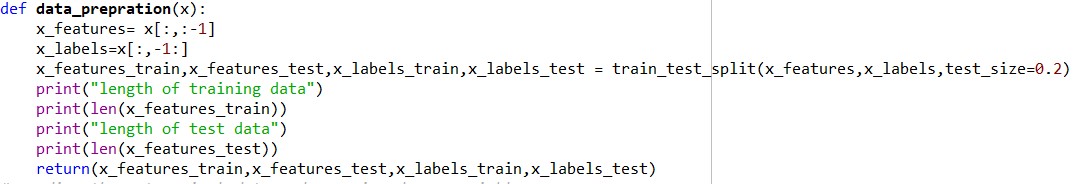




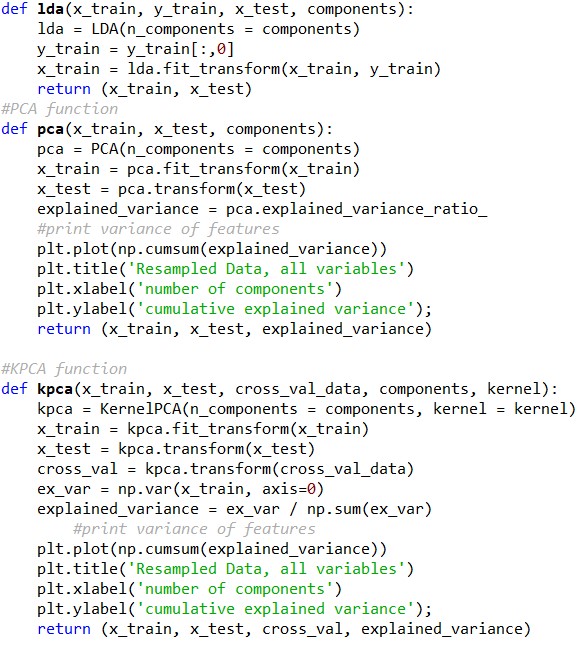
# 5) Categorical Data encoding Function



# 6) Splitting Data into Train and Test Sets Function



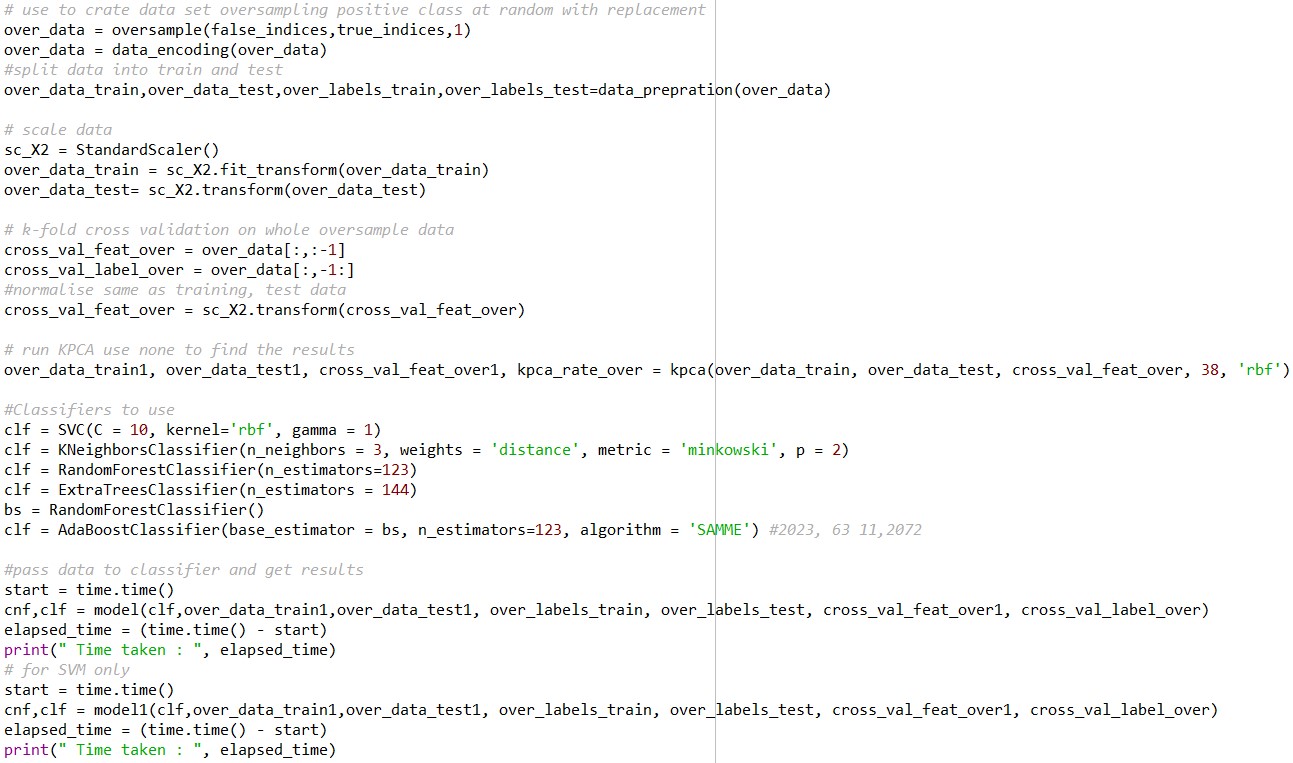
# 7) LDA, PCA, KPCA Functions



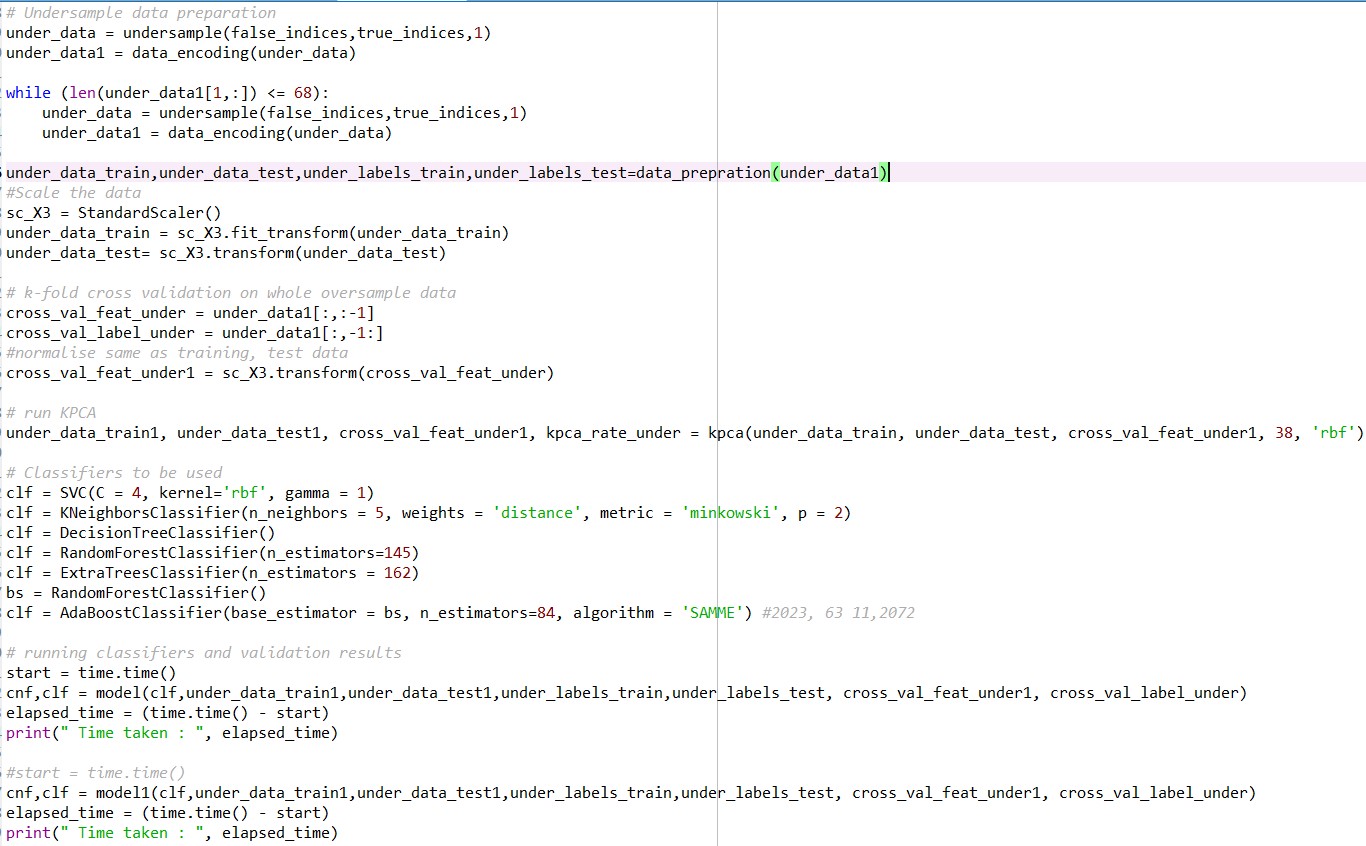
# 8) Running SMOTE Scenario



# 9) Running Random Over-sampling Scenario



# 10) Running Under-sample Scenario



# 11) Grid Search

