Predicting the Outcome of New Campaign basing on the Financial Details of Customer

This is submitted in partial fulfillment of the
requirements for the

Post Graduate Diploma in Data Science

By

A. Yeshwanth Srinath

18125760080

Under the guidance of

Mr.Mallikarjuna Doddamane
Faculty
Manipal Prolearn
Bangalore

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

Predicting the Outcome of New Campaign basing on the Financial Details of Customer

This is submitted in partial fulfillment of the
requirements for the

Post Graduate Diploma in Data Science

By

A. Yeshwanth Srinath

18125760080

Examiner 1	**Examiner 2**
Signature:	Signature:
Name:	Name:

MANIPAL ACADEMY OF HIGHER EDUCATION, MANIPAL

CERTIFICATE

This is to certify that the project work titled

Predicting the Outcome of New Campaign basing on the Financial Details of Customer

is a Bonafide record of the work done by

A. Yeshwanth Srinath

18125760080

In partial fulfillment of the requirements for the award of Post Graduate Diploma in Data Science under Manipal Academy of Higher Education and the same has not been submitted elsewhere for any kind of certification / recognition.

Mr. Mallikarjuna Doddamane

Faculty

Manipal Prolearn, Bangalore

1. ACKNOWLEDGEMENT

Acknowledgements:

I would like to express my deep gratitude to **Mr. Mallikarjuna Doddamane,**

my mentor for the project who clarified my understanding towards the data and guided me in data cleaning, data preparation and modelling.

I would also like to thank the project mid-term review panellists for their valuable feedback and suggestions that I incorporated in my project.

My thanks are also extended to professors who taught us subjects in

Data Science Diploma program specifically Exploratory Data Science,

Data Visualization and Machine Learning. The teachings in these subjects helped me to complete the project.

Finally, I wish to thank **Manipal Global Academy team** who supported with logistics and helped me in completing the project and Post Graduate Diploma in Data Science program on time.

2. Table of Contents

1. Acknowledgments ---------- 4
2. Table of contents with page number ---------- 5
3. Abstract ---------- 6
4. Introduction
   - 4.1. Motivation ---------- 7
   - 4.2. Project Goal ---------- 7
5. Project Description
   - 5.1. Data Description ---------- 8
   - 5.2. Data Limitations ---------- 8
   - 5.3. Dataset understanding ---------- 8
   - 5.4. Data Information ---------- 9
   - 5.5. Business Objective ---------- 9
   - 5.6. Tools and Technology ---------- 9-11
6. Exploratory Data Analysis
   - 6.1. Variable Identification ---------- 12
   - 6.2. Impact of Outliers on Dataset ---------- 13
   - 6.3. Data cleaning ---------- 13
7. Design
   - 7.1. Descriptive Statistical Analysis ---------- 14
   - 7.2. Data Visualization ---------- 15-19
8. Feature Selection
   - 8.1. Selecting features using Extra Tree Classifiers. ---------- 20
9. Label Encoding ---------- 22
10. Sampling
    - 10.1. Types of Sampling ---------- 23-24
    - 10.2. SMOTE for Handling Imbalanced Class ---------- 24-25
11. Model Building
    - 11.3. Random Forest Classifiers ---------- 26-28
    - 11.4. Cross Validation ---------- 28-31
    - 11.3. Decision Trees ---------- 32-33
    - 11.4. Extra Gradient Boosting ---------- 33-34
    - 11.5. Ada Boost Classifiers ---------- 34-35
    - 11.6. Final Synopsis of Algorithms Used ---------- 36
12. Conclusion ---------- 37
13. Reference ---------- 38

3. ABSTRACT

Basing on the given data by YES BANK, the financial history of the customer, predicting the outcome of the new campaign, including the outcomes of the previous campaign also. We were given the customer data including the fields like marital status, Education details, Job details, Loan details of the customer. We have to do the Exploratory Data Analysis by evaluating the measures like loan status, marital status, basing on their correlation with the campaign. We can also use the visualization techniques to get more insights in the given data. The basic approach for any data is, understand the data must be given the priority and have a good clarity in it. As we have to predict the effectiveness of the campaign, we have to make sure that feature selection should be done carefully. The next step is deciding the model, we can have a freedom to evaluate more than one models and get the best fitting model to the data. The next step is the hyperparameter tuning, which can also increase the effectiveness in our solution. The data given is the complete financial details of the customers, each row in the data represents a customer and each column says the details of the customer. The details include the loans taken by the customer, marital status, participation of the customer in the previous campaign and so on. We have to make an impactful decision basing on this data where a customer will participate in the new campaign or he will not.

4. INTRODUCTION

4.1 MOTIVATION:

My motivation towards this project is to know how the financial growth of person is affected by other factors and up to how much level is influenced on our Organisations Growth be it is positive influence or negative influence. How these factors are going to affect our organisation and main thing is are we going launch a product successfully or not or not – This is my moto.

4.2 PROJECT GOAL:

Focus of the project will be on the drivers from the data to get more insights and visualize and in data there is a target variable, so we can build classification model likes Decision Tree and Random Forest etc. and also explore many tuning methods for that models and get some good results

5.PROJECT DESCRIPTION

5.1 Data Description:

We were given the customer data including the fields like marital status, Education details, Job details, Loan details of the customer. We have to do the Exploratory Data Analysis by evaluating the measures like loan status , marital status, basing on their correlation with the campaign. We can also use the visualization techniques to get more insights in the given data. The data has some negative values in the fields like balance in the account and the days passed.

5.2 Data Limitations:

The Data is basically a complete financial year data of the customer. The data has some negative values in the fields like balance in the account and the days passed. The data is the set of reasonably clean record which were extracted from the data base of the yes bank.

5.3 Data Set Understanding:

The data set is mainly the records of the customers who have the financial relation with the Yes Bank. The data is mainly containing the output column as the 'Outcome', which is the outcome of the campaign which was going to be conducted

The Numerical Columns of the data set are:

```
In [17]: data._get_numeric_data().columns

Out[17]: Index(['age_in_years', 'balance_in_account', 'date', 'call_duration',
                'campaign_contacts', 'days_passed', 'previous_contact'],
               dtype='object')
```

The Categorical features of the data are:

```
In [12]: data_cat = data.select_dtypes(include=['object'])
         data_cat.columns

Out[12]: Index(['job_description', 'marital_status', 'education_details', 'has_default',
                'housing_status', 'previous_loan', 'phone_type', 'month_of_year',
                'poutcome_of_campaign', 'outcome'],
               dtype='object')
```

5.4 Data Information:

The data is consisting 31649 entries out which the seven numerical features and ten categorical features.

```
In [11]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31649 entries, 0 to 31648
Data columns (total 17 columns):
age_in_years            31649 non-null int64
job_description         31649 non-null object
marital_status          31649 non-null object
education_details       31649 non-null object
has_default             31649 non-null object
balance_in_account      31649 non-null int64
housing_status          31649 non-null object
previous_loan           31649 non-null object
phone_type              31649 non-null object
date                    31649 non-null int64
month_of_year           31649 non-null object
call_duration           31649 non-null int64
campaign_contacts       31649 non-null int64
days_passed             31649 non-null int64
previous_contact        31649 non-null int64
poutcome_of_campaign    31649 non-null object
outcome                 31649 non-null object
dtypes: int64(7), object(10)
memory usage: 4.1+ MB
```

5.5 Business Objective:

The project is mainly divided into two main business objectives as follows:

1. To understand the data and get the insights and visualize them and findings that are actionable in the community at large.
2. To explore Feature Selection techniques to identify the best features in the given list
3. Build the classification model likes Decision Tree and Random Forest to find wheather the new campaign will be effective for the bank or not.

5.6 Tools and Technology:

Anaconda:

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to Quickly download 1,500+ Python/R data science

packages. Manage libraries, dependencies, and environments with Conda. Develop and train machine learning and deep learning models with scikit-learn, TensorFlow
Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews.

Python:

Python is a very powerful programming language used for many different applications. Over time, the huge community around this open source language has created quite a few tools to efficiently work with Python. In recent years, a number of tools have been built specifically for data science. As a result, analysing data with Python has never been easier.
In this practical course, you will start from the very beginning, with basic arithmetic and variables, and learn how to handle data structures, such as Python lists, Numpy arrays, and Pandas Data Frames. Along the way, you'll learn about Python functions and control flow. Plus, you'll look at the world of data visualizations with Python and create your own stunning visualizations based on real data.

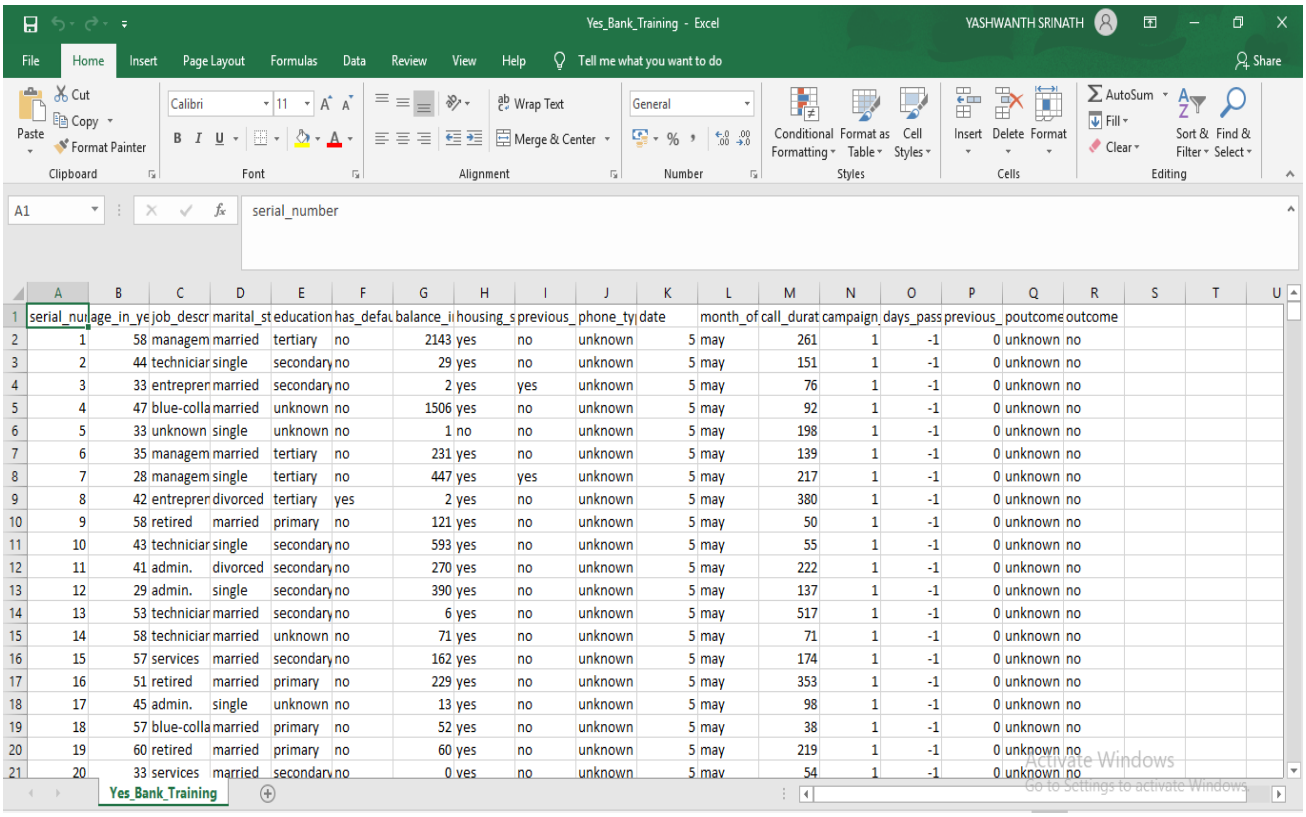
Pandas:

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Microsoft Excel:

Microsoft Excel is a spreadsheet developed by Microsoft for Windows, MacOS, Android and iOS. It features calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications. It has been a very widely applied spreadsheet for these platforms, especially since version 5 in 1993, and it has replaced Lotus 1-2-3 as the industry standard for spreadsheets. Excel forms part of Microsoft Office.Microsoft Excel has the basic features of all spreadsheets, using a grid of cells arranged in numbered rows and letter-

named columns to organize data manipulations like arithmetic operations. It has a battery of supplied functions to answer statistical, engineering and financial needs. In addition, it can display data as line graphs, histograms and charts, and with a very limited three-dimensional graphical display. It allows sectioning of data to view its dependencies on various factors for different perspectives (using pivot tables and the scenario manager). It has a programming aspect, Visual Basic for Applications, allowing the user to employ a wide variety of numerical methods, for example, for solving differential equations of mathematical physics, and then reporting the results back to the spreadsheet.



serial_nu	age_in_ye	job_descr	marital_st	education	has_defa	balance_i	housing_s	previous_	phone_ty	date	month_of	call_durat	campaign	days_pass	previous_	poutcome	outcome
1	58	managem	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
2	44	technicia	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
3	33	entrepren	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
4	47	blue-colla	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
5	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
6	35	managem	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	unknown	no
7	28	managem	single	tertiary	no	447	yes	yes	unknown	5	may	217	1	-1	0	unknown	no
8	42	entrepren	divorced	tertiary	yes	2	yes	no	unknown	5	may	380	1	-1	0	unknown	no
9	58	retired	married	primary	no	121	yes	no	unknown	5	may	50	1	-1	0	unknown	no
10	43	technicia	single	secondary	no	593	yes	no	unknown	5	may	55	1	-1	0	unknown	no
11	41	admin.	divorced	secondary	no	270	yes	no	unknown	5	may	222	1	-1	0	unknown	no
12	29	admin.	single	secondary	no	390	yes	no	unknown	5	may	137	1	-1	0	unknown	no
13	53	technicia	married	secondary	no	6	yes	no	unknown	5	may	517	1	-1	0	unknown	no
14	58	technicia	married	unknown	no	71	yes	no	unknown	5	may	71	1	-1	0	unknown	no
15	57	services	married	secondary	no	162	yes	no	unknown	5	may	174	1	-1	0	unknown	no
16	51	retired	married	primary	no	229	yes	no	unknown	5	may	353	1	-1	0	unknown	no
17	45	admin.	single	unknown	no	13	yes	no	unknown	5	may	98	1	-1	0	unknown	no
18	57	blue-colla	married	primary	no	52	yes	no	unknown	5	may	38	1	-1	0	unknown	no
19	60	retired	married	primary	no	60	yes	no	unknown	5	may	219	1	-1	0	unknown	no
20	33	services	married	secondary	no	0	yes	no	unknown	5	may	54	1	-1	0	unknown	no

6. EXPLORATORY DATA ANALYSIS

6.1 Variable Identification

First, identify **Predictor** (Input) and **Target** (output) variables. Next, identify the data type and category of the variables. In this section we try to find which type of variable it is. Whether it is dependent variable or independent variables. Then we find out the data types like the variable is character or numeric. After that we see it is categorical variable category or continuous variable category.

In our dataset, we did same steps for each column in the datasets. Our target variable is 'Outcome'. Then we find out the data type of variables like 'days passed' is numeric, 'Balance' is numeric, like we are checking each and every variable is continuous or categorical etc.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31649 entries, 0 to 31648
Data columns (total 17 columns):
age_in_years            31649 non-null int64
job_description         31649 non-null object
marital_status          31649 non-null object
education_details       31649 non-null object
has_default             31649 non-null object
balance_in_account      31649 non-null int64
housing_status          31649 non-null object
previous_loan           31649 non-null object
phone_type              31649 non-null object
date                    31649 non-null int64
month_of_year           31649 non-null object
call_duration           31649 non-null int64
campaign_contacts       31649 non-null int64
days_passed             31649 non-null int64
previous_contact        31649 non-null int64
poutcome_of_campaign    31649 non-null object
outcome                 31649 non-null object
dtypes: int64(7), object(10)
memory usage: 4.1+ MB
```

6.2 Impact of Outliers in the Data

Outlier is an observation that appears far away and diverges from an overall pattern in a sample. Outliers can drastically change the results of the data analysis and statistical modelling. There are numerous unfavourable impacts of outliers in the data set:

- It increases the error variance and reduces the power of statistical tests
- If the outliers are non-randomly distributed, they can decrease normality
- They can bias or influence estimates that may be of substantive interest

My Data:

There are not much outliers in the data, they are negligible and according to the domain we cannot randomly change the values or cap the values which are the outliers. The case is we need to know why the data was like that and then taking any decision would make it sensible

6.3 Data Cleaning

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model. It can lead to wrong prediction or classification. It is possible that there are problems with extraction process. In such cases, we should double-check for correct data with data guardians. Some hashing procedures can also be used to make sure data extraction is correct. Errors at data extraction stage are typically easy to find and can becorrected easily as well.

My Data

The Yes bank had not left any field empty in the data, but my data needed some cleaning as there are some negative values in the columns ‘Balance in account’ and ‘Days Passed’. I had given some good thought on it and had replaced all those values which are less than zero as zero, this was done because the negative values will show impact on the model when built. So, considering balance less than zero to be zero. I had done all the negative values to zeros.

7. DESIGN

7.1 Descriptive Statistical Analysis:

Descriptive statistics are used to describe the basic features of the data in a study. They provide simple summaries about the sample and the measures. Together with simple graphics analysis, they form the basis of virtually every quantitative analysis of data.

Summary of the Data:

Data that summarize all observations in a category are called summarized data. The summary could be the sum of the observations, the number of occurrences, their mean value, and so on. When the summary is the number of occurrences, this is known as frequency data.

```
data.describe()
```

	age_in_years	balance_in_account	date	call_duration	campaign_contacts	days_passed	previous_contact
count	31649.000000	31649.000000	31649.000000	31649.000000	31649.000000	31649.000000	31649.000000
mean	41.111820	1293.382540	16.597997	252.408196	3.062372	10.819931	0.208537
std	9.597652	2961.185138	8.586414	262.344981	3.510245	48.086849	1.946336
min	19.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	47.000000	9.000000	98.000000	1.000000	-1.000000	0.000000
50%	40.000000	397.000000	18.000000	171.000000	2.000000	-1.000000	0.000000
75%	49.000000	1328.000000	23.000000	305.000000	3.000000	-1.000000	0.000000
max	94.000000	98417.000000	31.000000	4918.000000	63.000000	335.000000	275.000000

7.2 Data Visualization

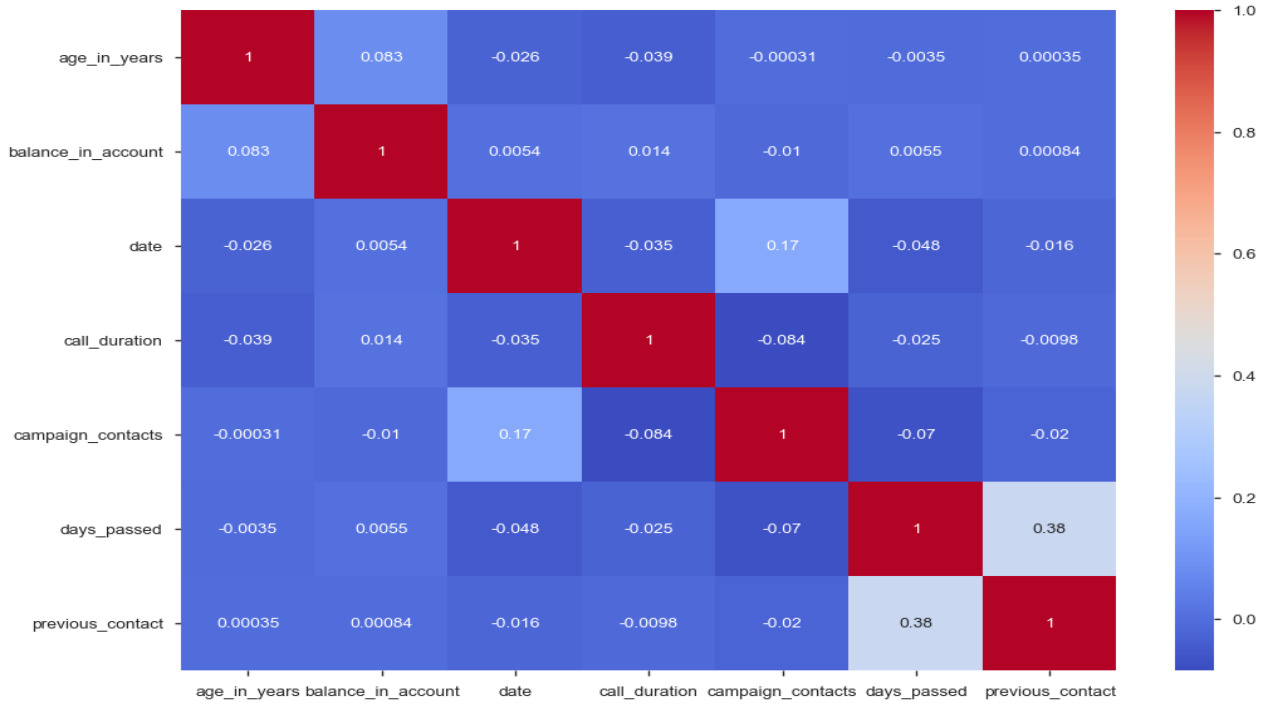
Correlations:

Correlation Matrix is basically a covariance matrix. Also known as the auto-covariance matrix, dispersion matrix, variance matrix, or variance-covariance matrix. It is a matrix in which i-j position defines the correlation between the i^{th} and j^{th} parameter of the given data-set.
When the data points follow a roughly straight-line trend, the variables are said to have an approximately linear relationship. In some cases, the data points fall close to a straight line, but more often there is quite a bit of variability of the points around the straight-line trend. A summary measure called the correlation describes the strength of the linear association. Correlation summarizes the strength and direction of the linear (straight-line) association between two quantitative variables. Denoted by r, it takes values between -1 and +1. A positive value for r indicates a positive association, and a negative value for r indicates a negative association. The closer r is to 1 the closer the data points fall to a straight line , thus, the linear association is stronger. The closer r is to 0, making the linear association weaker.

```
]: data.corr()
```

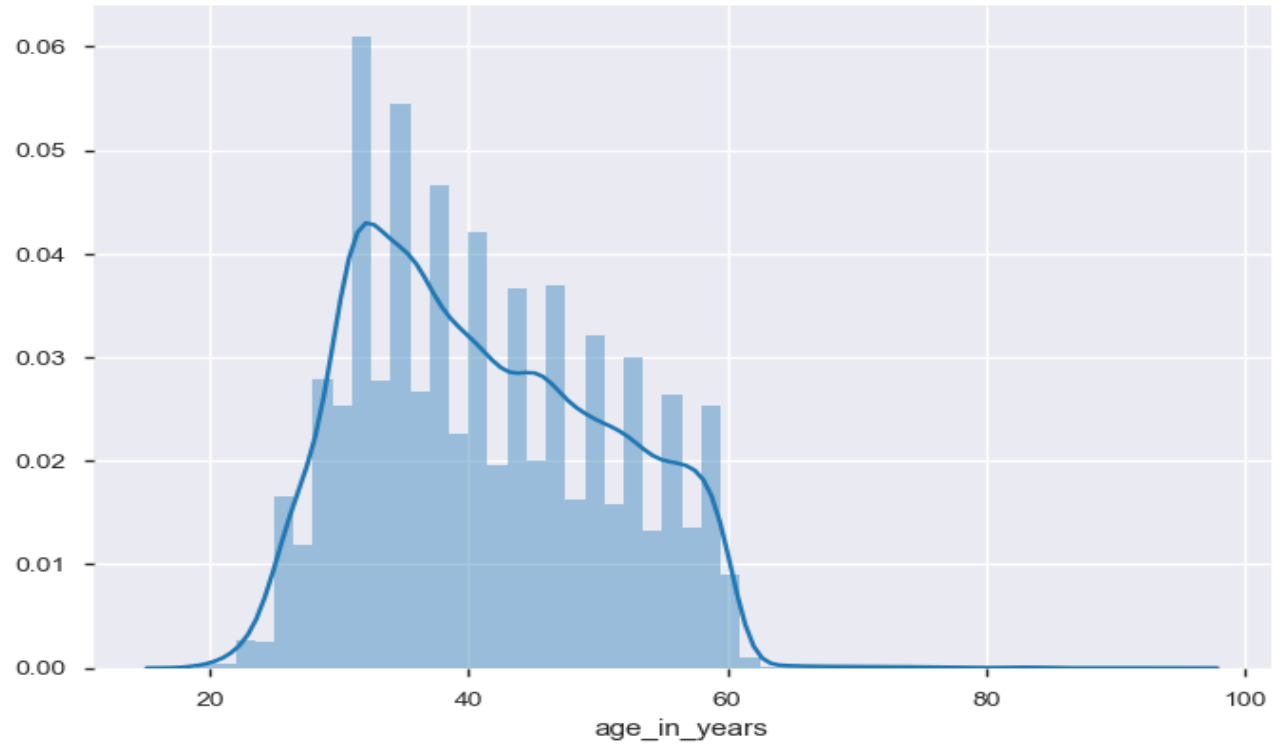
]:

	age_in_years	balance_in_account	date	call_duration	campaign_contacts	days_passed	previous_contact
age_in_years	1.000000	0.083122	-0.025832	-0.038859	-0.000308	-0.003469	0.000350
balance_in_account	0.083122	1.000000	0.005410	0.013550	-0.010280	0.005534	0.000839
date	-0.025832	0.005410	1.000000	-0.034967	0.169599	-0.048191	-0.016339
call_duration	-0.038859	0.013550	-0.034967	1.000000	-0.083680	-0.025242	-0.009836
campaign_contacts	-0.000308	-0.010280	0.169599	-0.083680	1.000000	-0.069951	-0.020251
days_passed	-0.003469	0.005534	-0.048191	-0.025242	-0.069951	1.000000	0.380066
previous_contact	0.000350	0.000839	-0.016339	-0.009836	-0.020251	0.380066	1.000000



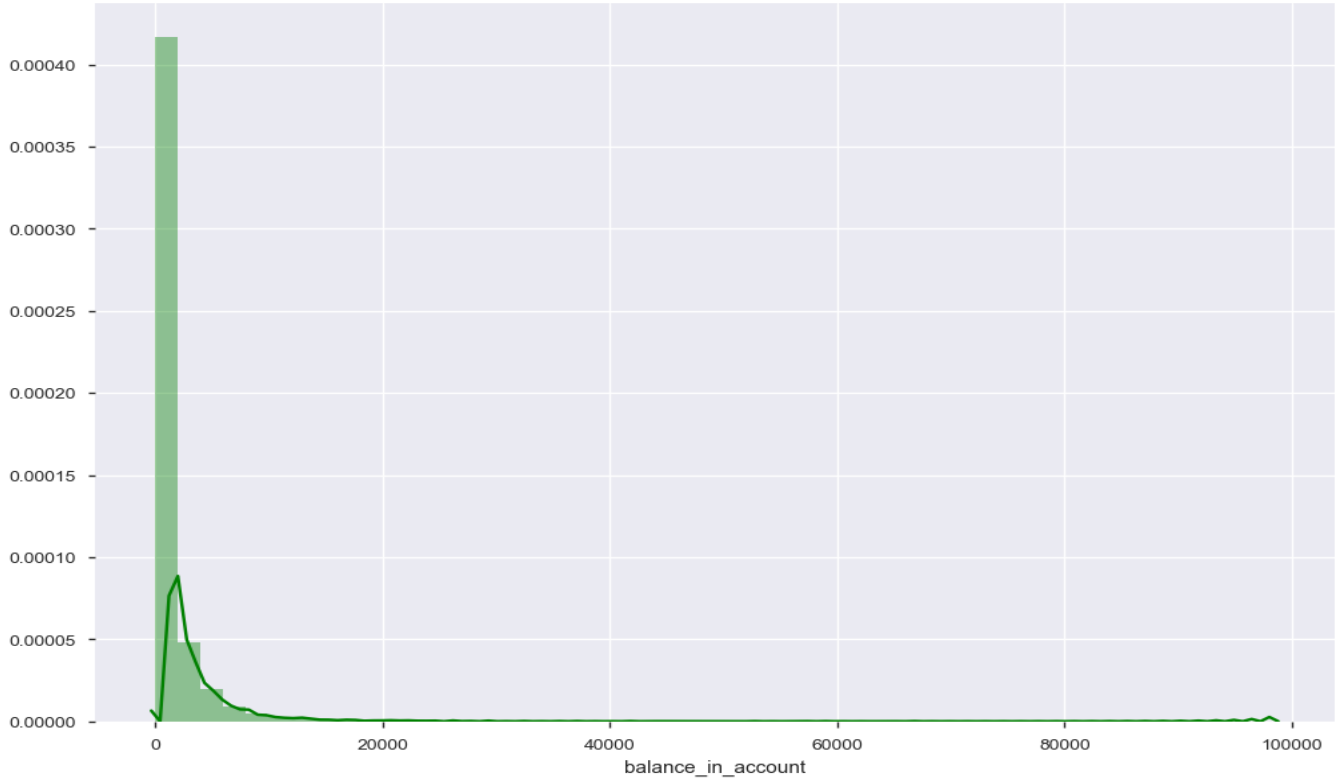
Distribution of Age in Years of the customers:

Since the data is randomly pulled out of bank database of those customers who are having previous financial history associated with the bank, the age may have a bit of variations. The customers with the age ranging between 20 to 30 are basically more in number in the given data. This is what I observed in the below graph

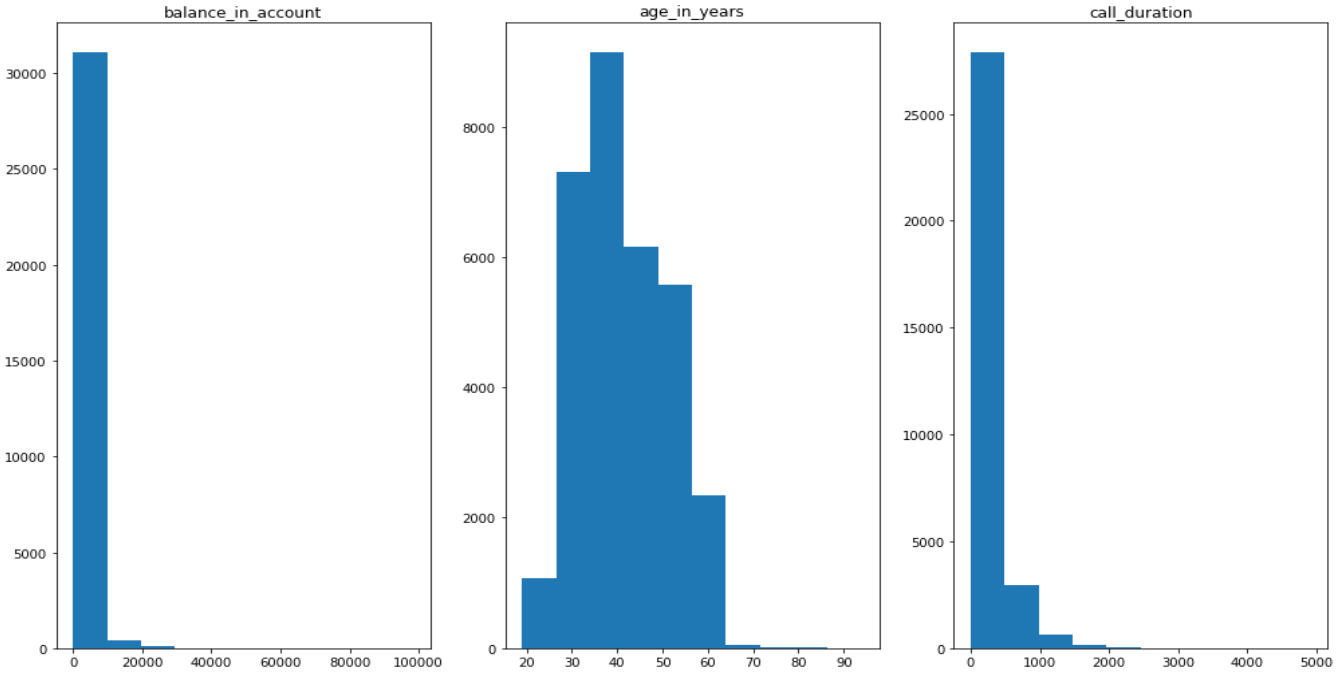


Distribution of Balance in the Account:

The Balance in account distribution is seen to more tending in favor of the zeros, The customers with balance with zeros are more in number. This might also be the result of the data cleaning because of the negative values which will affect the model performance were capped to zeros, this might cause the distribution to be changed.

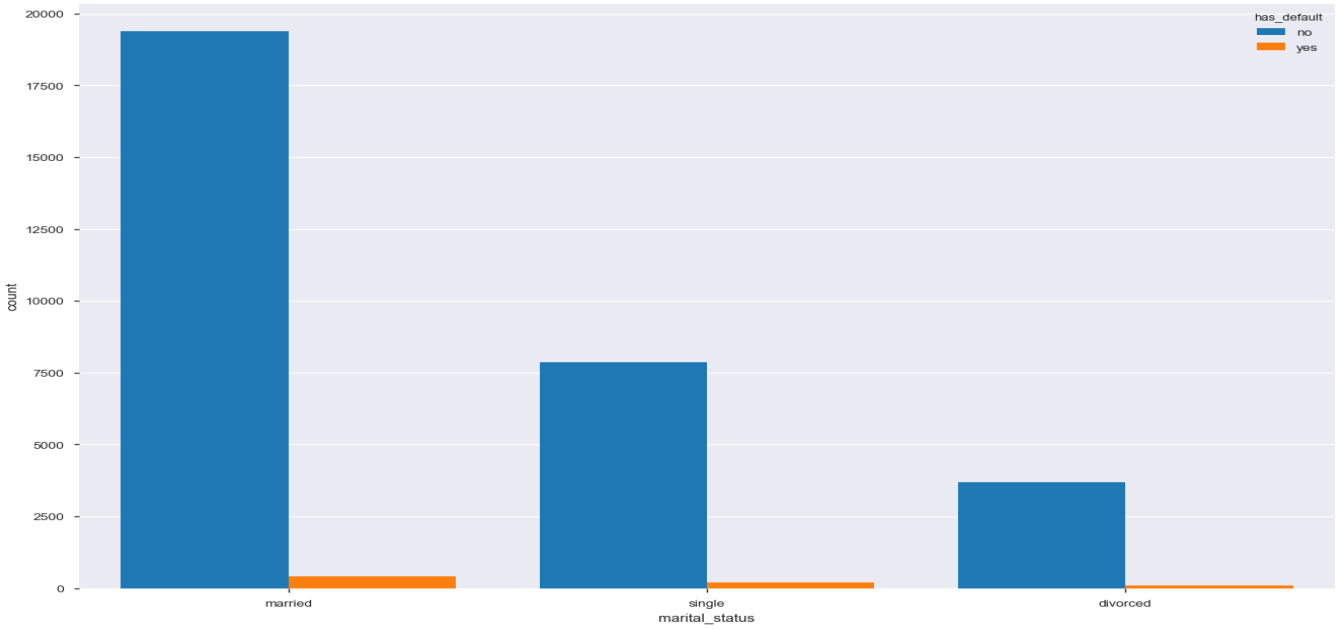


Distributions of Age and Call Durations:

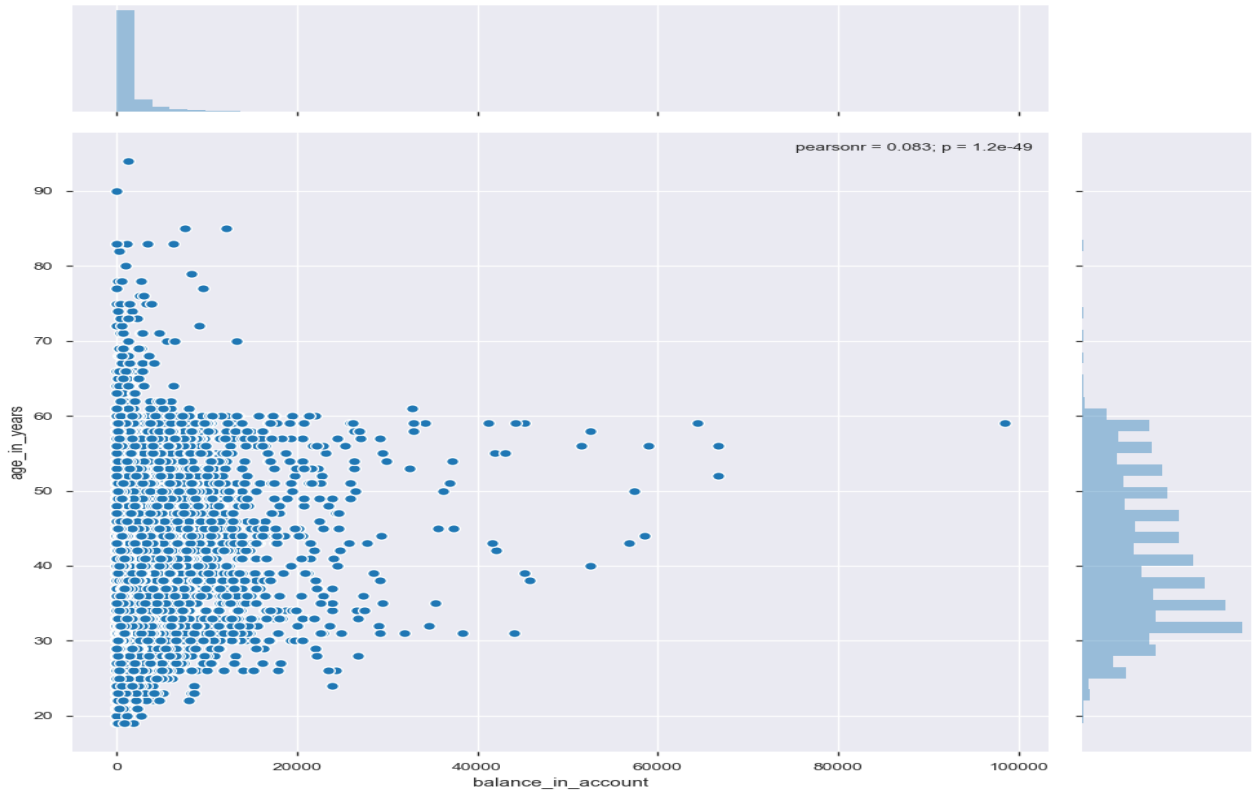


Plot to show Marital status in relation with Having Default:

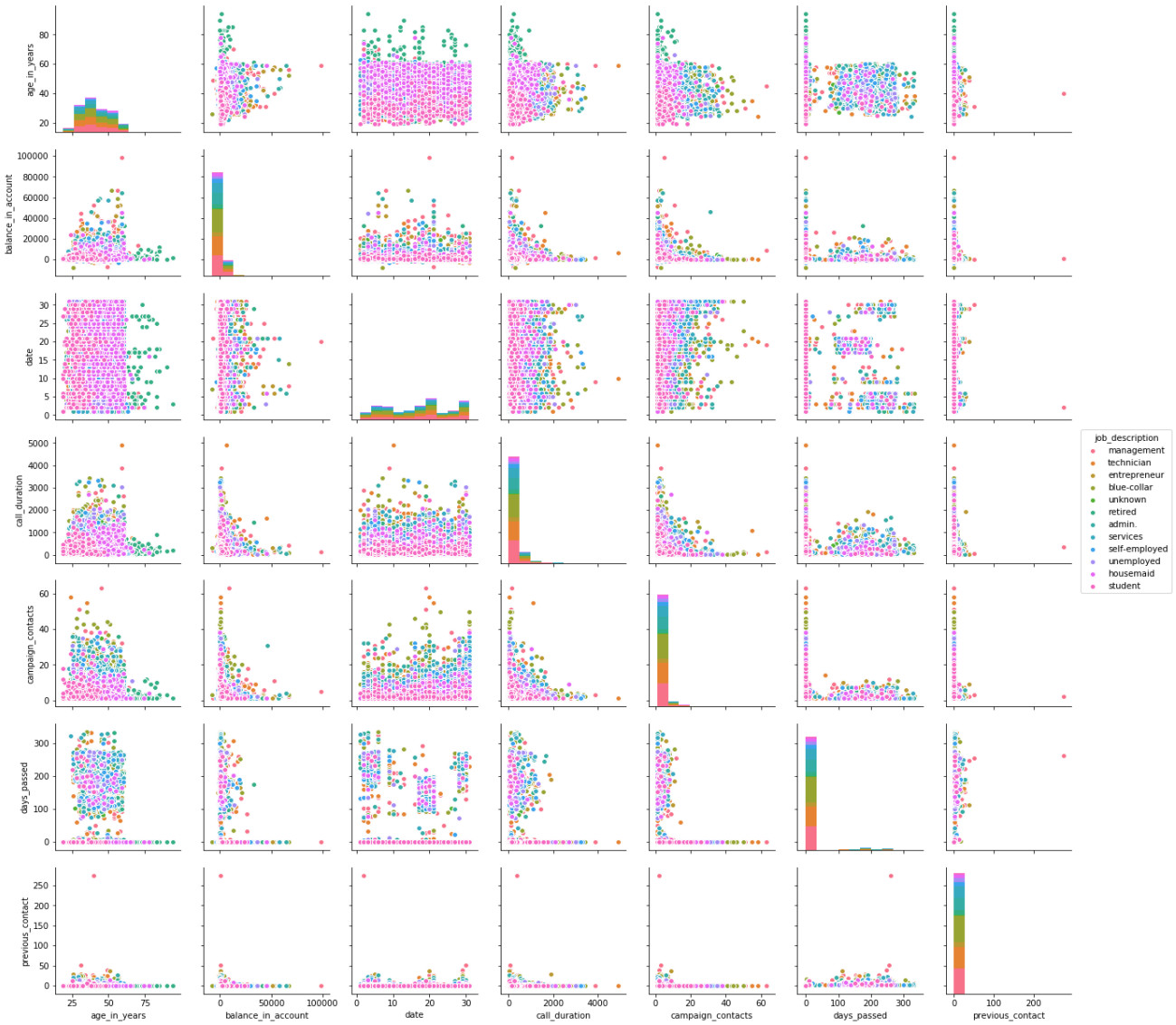
We can clearly observe that the Married are the people who are having less defaults and they are also the ones who are having the lesser defaults.



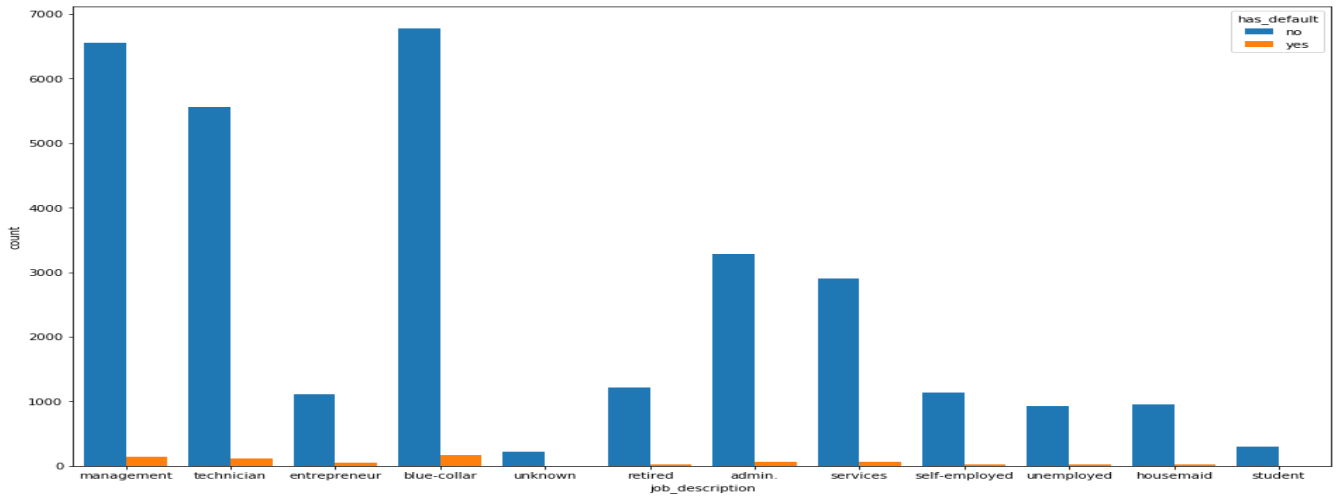
Plot to show correlation between Age and Balance:



Pair Plot Showing the Distributions:



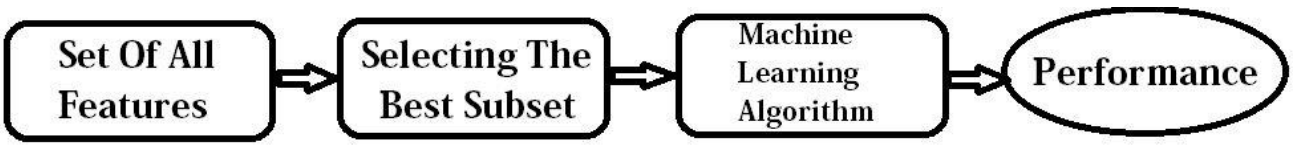
Plot showing the Defaults with respect to Educational Status:



8. FEATURE SELECTION

Feature selection is the method of reducing data dimension while doing predictive analysis. One major reason is that machine learning follows the rule of “garbage in-garbage out” and that is why one needs to be very concerned about the data that is being fed to the model.In this article, we will discuss various kinds of feature selection techniques in machine learning and why they play an important role in machine learning tasks.

Filter Method



This method uses the variable ranking technique in order to select the variables for ordering and here, the selection of features is independent of the classifiers used. By ranking, it means how much useful and important each feature is expected to be for classification. It basically selects the subsets of variables as a pre-processing step independently of the chosen predictor. In filtering, the ranking method can be applied before classification for filtering the less relevant features. It carries out the feature selection task as a pre-processing step which contains no induction algorithm.

Wrapper Method

The Wrapper Methodology was made famous by researchers Ron Kohavi and George H. John in the year 1997. This method utilises the learning machine of interest as a black box to score subsets of variables according to their predictive power. In the above figure, in a supervised machine learning, the induction algorithm is depicted with a set of training instances, where each instance is described by a vector of feature values and a class label. The induction algorithm which is also considered as the black box is used to induce a classifier which is useful in classifying. In the wrapper approach, the feature subset selection algorithm exists as a wrapper around the induction algorithm. One of the main drawbacks of this technique is the mass of computations required to obtain the feature subset.

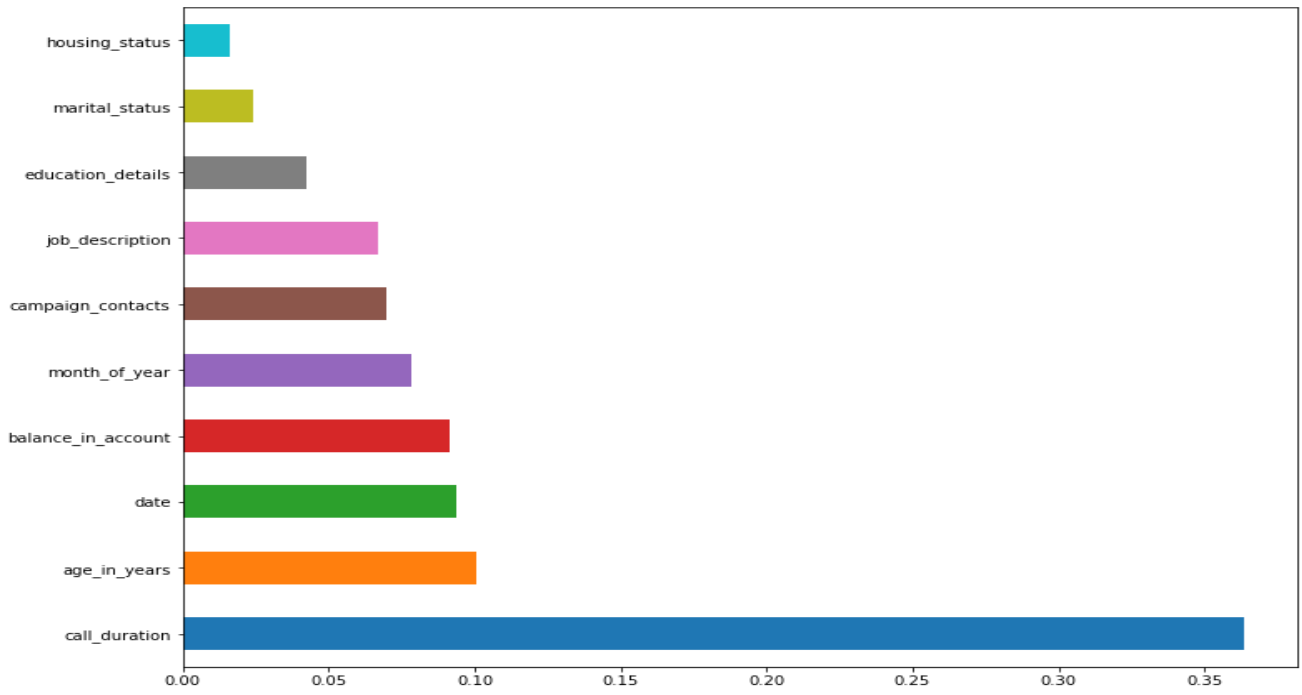
8.1 Selecting Features using Extra Tree Classifier

You can get the feature importance of each feature of your dataset by using the feature importance property of the model. Feature importance gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable. Feature importance is an inbuilt class that comes with Tree Based Classifiers, we will be using Extra Tree Classifier for extracting the top 10 features for the dataset.

```
df1 = data1.drop(['outcome'],axis=1)
y = data1[['outcome']]
X = df1#independent columns
y = y
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
fig = plt.figure(figsize = (12,10))
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

```
C:\Users\user\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default va
change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\user\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: DataConversionWarning: A column-vect
d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

```
[0.10057477 0.06692866 0.02417951 0.04268398 0.00505754 0.09143149
 0.01644883 0.01040916 0.01541732 0.09380069 0.07846902 0.36364876
 0.0695642  0.00779239 0.00703047 0.00656322]
```



9. LABEL ENCODING

In machine learning, we usually deal with datasets which contains multiple labels in one or more than one columns. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labelled in words.

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Example:

Suppose we have a column Height in some dataset.

Height
Tall
Medium
Short

After applying label encoding, the Height column is converted into:

Height
0
1
2

where 0 is the label for tall, 1 is the label for medium and 2 is label for short height.

```
In [13]: from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         le.fit(data_cat['poutcome_of_campaign'])

Out[13]: LabelEncoder()

In [24]: list(le.classes_)

Out[24]: ['failure', 'other', 'success', 'unknown']

In [25]: le.transform(data_cat['poutcome_of_campaign'])

Out[25]: array([3, 3, 3, ..., 0, 3, 3])

In [26]: list(le.inverse_transform([0,1,2,3]))

Out[26]: ['failure', 'other', 'success', 'unknown']

In [14]: data1 = data.apply(LabelEncoder().fit_transform)
```

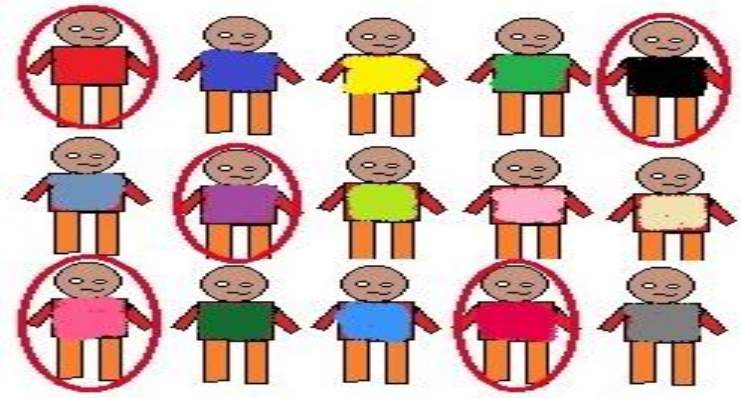
10. SAMPLING

Sampling helps a lot in research. It is one of the most important factors which determines the accuracy of your research/survey result. If anything goes wrong with your sample then it will be directly reflected in the final result. There are lot of techniques which help us to gather sample depending upon the need and situation.

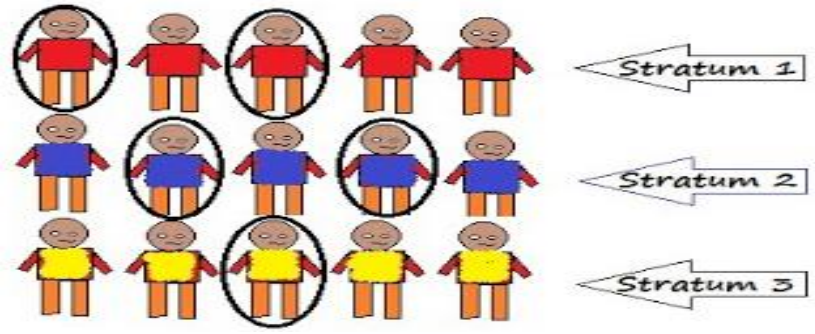
10.1 Types of Sampling:

Simple Random Sampling:

Every element has an equal chance of getting selected to be the part sample. It is used when we don't have any kind of prior information about the target population.



Stratified Sampling: This technique divides the elements of the population into small subgroups (strata) based on the similarity in such a way that the elements within the group are homogeneous and heterogeneous among the other subgroups formed. And then the elements are randomly selected from each of these strata. We need to have prior information about the population to create subgroups.



Stratified Sampling

Random oversampling:

Random Oversampling involves supplementing the training data with multiple copies of some of the minority classes. Oversampling can be done more than once (2x, 3x, 5x, 10x, etc.) This is one of the earliest proposed methods, that is also proven to be robust.Instead of duplicating every sample in the minority class, some of them may be randomly chosen with replacement.

10.2 SMOTE Technique for Handling Imbalance in class:

When one class of data is the underrepresented minority class in the data sample, over sampling techniques maybe used to duplicate these results for a more balanced amount of positive results in training. Over sampling is used when the amount of data collected is insufficient. A popular over sampling technique is SMOTE (Synthetic Minority Over-sampling Technique), which creates synthetic samples by randomly sampling the characteristics from occurrences in the minority class.

Conversely, if a class of data is the overrepresented majority class, under sampling may be used to balance it with the minority class. Under sampling is used when the amount of collected data is sufficient. Common methods of under sampling include cluster centroids and Tomek links, both of which target potential overlapping characteristics within the collected data sets to reduce the amount of majority data.

In both over sampling and under sampling, simple data duplication is rarely suggested. Generally, over sampling is preferable as under sampling can result in the loss of important data. Under sampling is suggested when the amount of data collected is larger than ideal and can help data mining tools to stay within the limits of what they can effectively process.

What is an Imbalanced Dataset?

Imagine, you have two categories in your dataset to predict—Category-A and Category-B. When Category-A is higher than Category-B or vice versa, you have a problem of imbalanced dataset.

```
data['outcome'].value_counts()
```

```
no     29809
yes     1840
Name: outcome, dtype: int64
```

SMOTE is an over-sampling method. What it does is, it creates synthetic (not duplicate) samples of the minority class. Hence making the minority class equal to the majority class. SMOTE does this by selecting similar records and altering that record one column at a time by a random amount within the difference to the neighbouring records.

We will be diving into python to see how this works. If you want to read more on SMOTE, here is an original research paper titled: “SMOTE: Synthetic Minority Over-sampling Technique” written in 2002.

11.MODEL BUILDING

Since, The data is having the output column ‘outcome’ which is the categorical column. The plan is to use the Classification algorithms.

Libraries Used (Python):

- pandas
- matplotlib
- numpy
- sklearn
- seaborn
- DecisionTreeClassifier
- RandomForestClassifier
- AdaboostClassifier
- XGBoost Classifier
- Cross Validation
- Grid Search CV

11.1 Random Forest Classifiers:

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data1,test_size=0.2,random_state=100)
train_y= train['outcome']
test_y= test['outcome']
#Since in the input data we do not need the outcome as we are predicting the outcome so we drop out the outcome column
train_x = train.drop('outcome',axis=1)
test_x = test.drop('outcome',axis=1)



rf = RandomForestClassifier(random_state=100) #Use this for Random classifier
rf.fit(train_x,train_y)
```

```
C:\Users\user\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimat
change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=100, verbose=0, warm_start=False)
```

Prediction of Random Forest:

In [23]:
```
from sklearn.metrics import accuracy_score
accuracy_rf_basic = accuracy_score(test_y,test_pred)
print(accuracy_rf_basic)
```

```
0.9448657187993681
```

In [24]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(test_y,test_pred)
```

Out[24]:
```
array([[5887,   87],
       [ 262,   94]], dtype=int64)
```

In [25]:
```
from sklearn.metrics import classification_report
tn, fp, fn, tp = confusion_matrix(test_y,test_pred).ravel()
print('TruePositve %d' %tp)
print('TrueNegetive %d' %tn)
print('FalsePosiive %d' %fp)
print('FalseNegetive %d' %fn)
```

```
TruePositve 94
TrueNegetive 5887
FalsePosiive 87
FalseNegetive 262
```

In [26]:
```
print(classification_report(test_y,test_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      5974
           1       0.52      0.26      0.35       356

   micro avg       0.94      0.94      0.94      6330
   macro avg       0.74      0.62      0.66      6330
weighted avg       0.93      0.94      0.94      6330
```

Hyper Parameter Tuning Using GridSearchCv:

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyper parameter values specified. If you work with ML, you know what a nightmare it is to stipulate values for hyper parameters. There are libraries that have been implemented, such as GridSearchCV of the sklearn library, in order to automate this process and make life a little bit easier for ML enthusiasts.

HyperParameter Tuning Random Forest for Imbalanced Class

```
params = {'criterion':['gini','entropy'],
          'n_estimators':[10,15,20,25,30],
          'min_samples_leaf':[1,2,3],
          'min_samples_split':[3,4,5,6,7],
          'random_state':[123],
          'n_jobs':[-1]}
```

```
rf_tun = GridSearchCV(rf, param_grid=params,cv=5)

rf_tun.fit(train_x,train_y)

print("Best Hyper Parameters:\n",rf_tun.best_params_)

prediction_rf_tun=rf_tun.predict(test_x)

from sklearn import metrics
```

```
Best Hyper Parameters:
 {'criterion': 'gini', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 30, 'n_jobs': -1, 'random_state': 123}
```

11.2 Cross Validation:

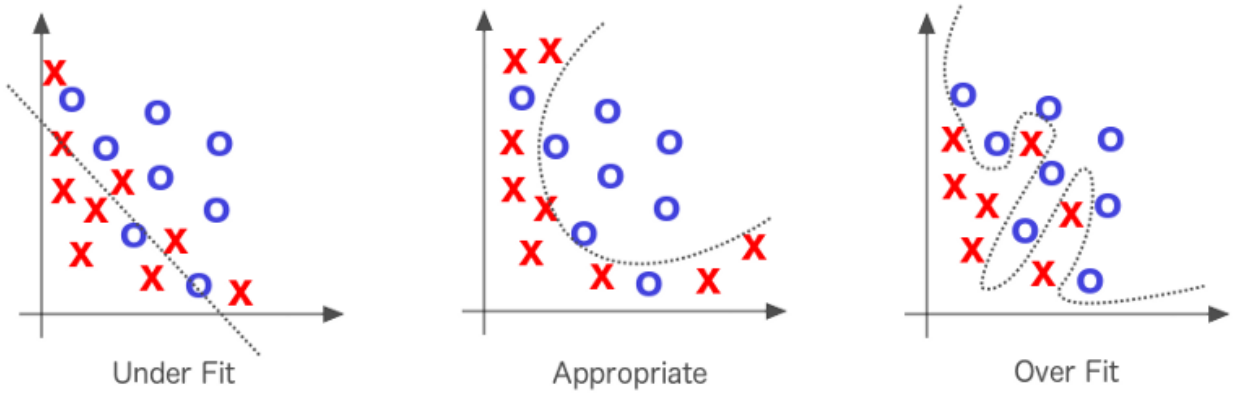
One Solution to Overfitting: Cross-Validation

Cross-validation is one solution to overfitting. The idea is that once we have identified our best combination of parameters (in our case time and route) we test the performance of that set of parameters in a different context. Therefore, we may want to test on Tue and Thu as well to ensure that our choices work for those days as well.

Overfit Model**:** Overfitting occurs when a statistical model or machine learning algorithm captures the noise of the data. Intuitively, overfitting occurs when the model or the algorithm fits the data too well.

Overfitting a model result in good accuracy for training data set but poor results on new data sets. Such a model is not of any use in the real world as it is not able to predict outcomes for new cases.

Underfit Model: Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. Intuitively, underfitting occurs when the model or the algorithm does not fit the data well enough. Underfitting is often a result of an excessively simple model. By simple we mean that the missing data is not handled properly, no outlier treatment, removing of irrelevant features or features which do not contribute much to the predictor variable.



How to tackle Problem of Overfitting:

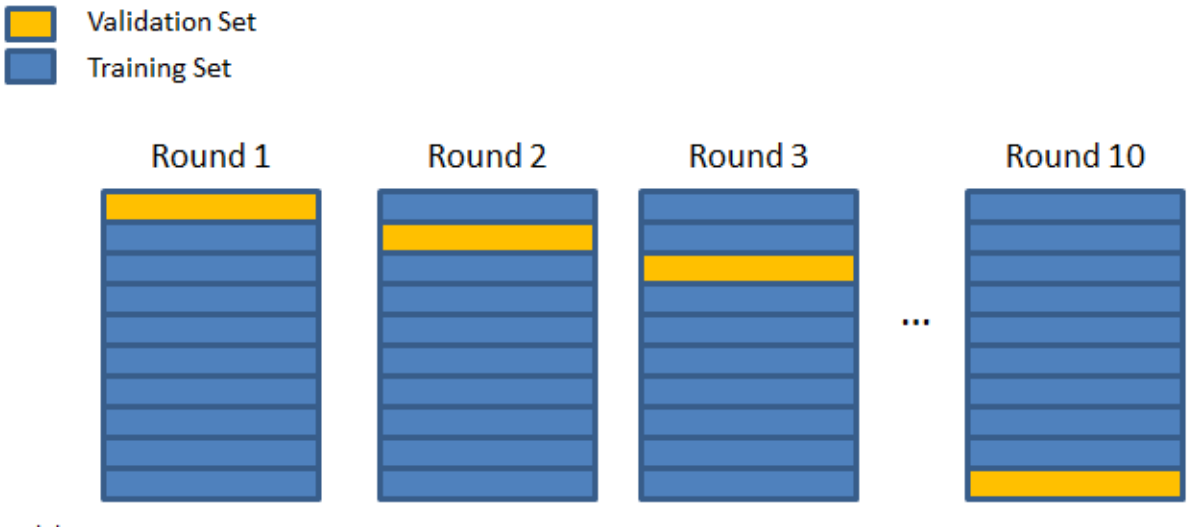
The answer is Cross Validation

A key challenge with overfitting, and with machine learning in general, is that we can't know how well our model will perform on new data until we actually test it.

To address this, we can split our initial dataset into separate training and test subsets.

There are different types of Cross Validation Techniques but the overall concept remains the same,

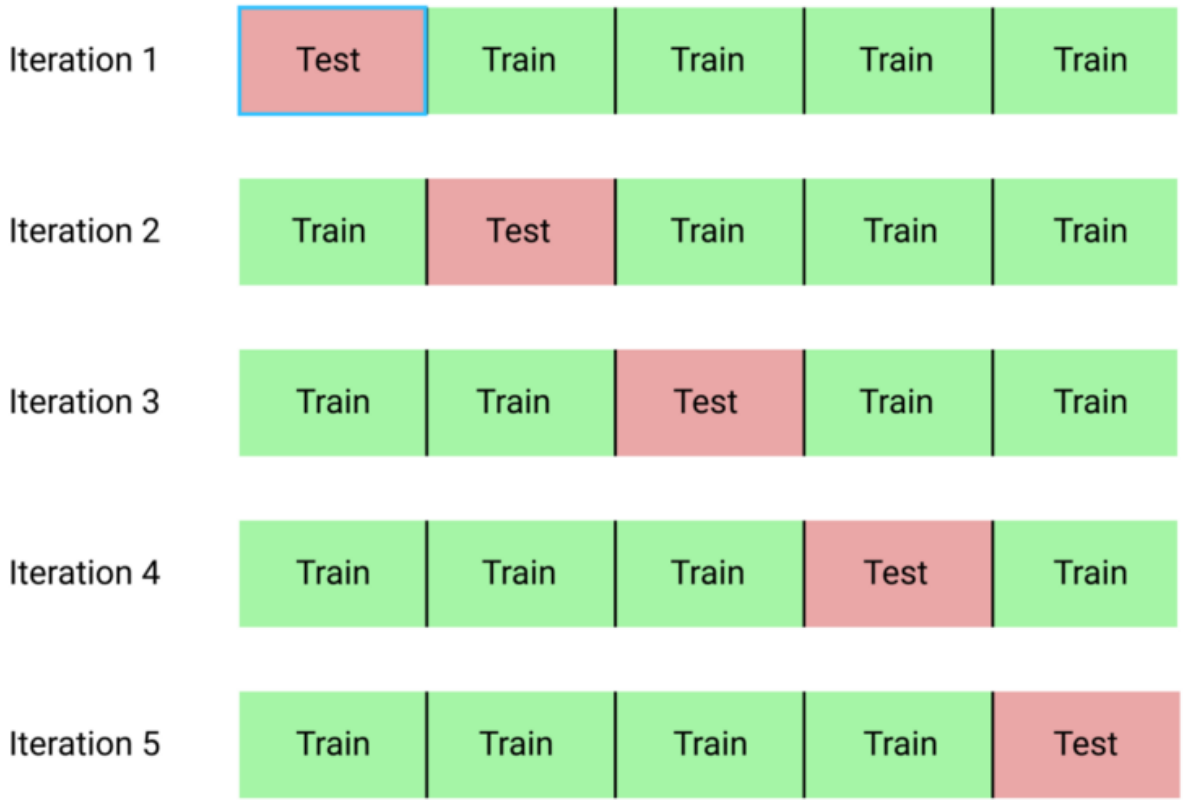
- To partition the data into a number of subsets
- Hold out a set at a time and train the model on remaining set
- Test model on hold out set



Types of Cross Validation:

- K-Fold Cross Validation
- Stratified K-fold Cross Validation
- Leave One Out Cross Validation

k-Fold Cross Validation:



The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

If k=5 the dataset will be divided into 5 equal parts and the below process will run 5 times, each time with a different holdout set.

1. Take the group as a holdout or test data set
2. Take the remaining groups as a training data set
3. Fit a model on the training set and evaluate it on the test set
4. Retain the evaluation score and discard the model

At the end of the above process Summarize the skill of the model using the sample of model evaluation scores.

How to decide the value of k?

The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.

A value of k=10 is very common in the field of applied machine learning, and is recommend if you are struggling to choose a value for your dataset.

If a value for k is chosen that does not evenly split the data sample, then one group will contain a remainder of the examples. It is preferable to split the data sample into k groups with the same number of samples, such that the sample of model skill scores are all equivalent.

Application of K-Fold Cross Validation for Random Forest

In [40]:
```
from sklearn.model_selection import cross_val_score
rand = RandomForestClassifier(random_state=42) #Use this for Random classifier
rand.fit(X_train,y_train)
accuracy = cross_val_score(rand,X_train,y_train, scoring='accuracy', cv = 10)
print(accuracy)
print("Accuracy of Model with Cross Validation is:",accuracy.mean() * 100)
```

```
C:\Users\user\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The
change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

[0.84686347 0.95790003 0.96041597 0.94984904 0.75813485 0.92770882
 0.82841328 0.95739685 0.82304596 0.68204698]
Accuracy of Model with Cross Validation is: 86.91775258966746
```

In [41]:
```
cv_rf = accuracy.mean() * 100
cv_rf
```

Out[41]:
```
86.91775258966746
```

11.3 Decision Trees:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. Decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Decision Tree Predictions:

[42]:
```
model = DecisionTreeClassifier(criterion='gini', splitter='best') #Use This for Decision Tree
model.fit(train_x,train_y)
print('ACCURACY SCORE:',accuracy_score(test_y,test_pred))
print(classification_report(test_y,test_pred))
```

```
ACCURACY SCORE: 0.9448657187993681
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      5974
           1       0.52      0.26      0.35       356

   micro avg       0.94      0.94      0.94      6330
   macro avg       0.74      0.62      0.66      6330
weighted avg       0.93      0.94      0.94      6330
```

HyperParameter Tuning Decision Tree:

```
dec = DecisionTreeClassifier(criterion='gini', splitter='best')
```

```
params = {'max_features': ['auto', 'sqrt', 'log2'],
          'min_samples_split': [2,4,5,6,9,10,14,15],
          'min_samples_leaf':[1,2,3,4,7,9,10,15],
          'random_state':[123,100,42]}
```

```
dec_tun = GridSearchCV(dec, param_grid=params, cv =5)
dec_tun.fit(train_x,train_y)
print("Best Hyper Parameters:",dec_tun.best_params_)
prediction=dec_tun.predict(test_x)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(prediction,test_y))
tn, fp, fn, tp = confusion_matrix(prediction,test_y).ravel()
print('TruePositve %d' %tp)
print('TrueNegetive %d' %tn)
print('FalsePosiive %d' %fp)
print('FalseNegetive %d' %fn)
```

```
Best Hyper Parameters: {'max_features': 'auto', 'min_samples_leaf': 15, 'min_samples_split': 2, 'random_state': 100}
Accuracy: 0.94565560821485
TruePositve 116
TrueNegetive 5870
FalsePosiive 240
FalseNegetive 104
```

Decision Tree Cross Validation:

```
In [59]: from sklearn.model_selection import cross_val_score
         des = DecisionTreeClassifier(criterion='gini', splitter='best') #Use This for Decision Tree
         des.fit(X_train,y_train)
         accuracy = cross_val_score(des,X_train,y_train, scoring='accuracy', cv = 10)
         print(accuracy)

         [0.82941966 0.93274069 0.82472325 0.89265347 0.58570949 0.82723918
          0.74203287 0.87990607 0.64793693 0.64026846]

In [60]: print("Accuracy of Model with Cross Validation is:",accuracy.mean() * 100)

         Accuracy of Model with Cross Validation is: 78.02630079992076

In [61]: cv_dt = accuracy.mean() * 100
         print(cv_dt)

         78.02630079992076
```

11.4 XGBOOST:

XGBoost stands for “Extreme Gradient Boosting”, where the term “Gradient Boosting” originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. This is a tutorial on gradient boosted trees, and most of the content is based on these slides by Tianqi Chen, the original author of XGBoost.

The gradient boosted trees has been around for a while, and there are a lot of materials on the topic. This tutorial will explain boosted trees in a self-contained and principled way using the elements of supervised learning. We think this explanation is cleaner, more formal, and motivates the model formulation used in XGBoost.

XG Boost Prediction:

```
from xgboost.sklearn import XGBClassifier
Xg = XGBClassifier(learning_rate=0.1, n_estimators=140, max_depth=5,
                   min_child_weight=3, gamma=0.2, subsample=0.6, colsample_bytree=1.0,
                   objective='binary:logistic', nthread=4, scale_pos_weight=1, seed=27)
```

```
Xg.fit(train_x,train_y)

pred_xg=Xg.predict(test_x)

score_xg_b=accuracy_score(test_y,pred_xg)
print("accuracy:   %0.3f" % score_xg_b)
from sklearn.metrics import classification_report
print(classification_report(test_y,pred_xg))
```

```
accuracy:   0.950
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      5974
           1       0.59      0.35      0.44       356

   micro avg       0.95      0.95      0.95      6330
   macro avg       0.78      0.67      0.70      6330
weighted avg       0.94      0.95      0.94      6330
```

XG Boost Hyper Parameter Tuning:

```
In [44]: parameters = {'nthread':[4], #when use hyperthread, xgboost may become slower
                       'objective':['binary:logistic'],
                       'learning_rate': [0.05], #so called `eta` value
                       'max_depth': [6],
                       'min_child_weight': [11],
                       'silent': [1],
                       'subsample': [0.8],
                       'colsample_bytree': [0.7],
                       'n_estimators': [1000], #number of trees, change it to 1000 for better results
                       'missing':[-999],
                       'seed': [1337]}

In [45]: CV_rfc = GridSearchCV(estimator=Xg, param_grid=parameters, cv= 5)
         CV_rfc.fit(train_x, train_y)

Out[45]: GridSearchCV(cv=5, error_score='raise-deprecating',
                estimator=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                colsample_bynode=1, colsample_bytree=1.0, gamma=0.2,
                learning_rate=0.1, max_delta_step=0, max_depth=5,
                min_child_weight=3, missing=None, n_estimators=140, n_jobs=1,
                nthread=4, objective='binary:logistic', random_state=0, reg_alpha=0,
                reg_lambda=1, scale_pos_weight=1, seed=27, silent=None,
                subsample=0.6, verbosity=1),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'nthread': [4], 'objective': ['binary:logistic'], 'learning_rate': [0.05], 'max_depth': [6], 'min_child_weig
         ht': [11], 'silent': [1], 'subsample': [0.8], 'colsample_bytree': [0.7], 'n_estimators': [1000], 'missing': [-999], 'seed': [13
         37]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
```


XG Boost Cross Validation:

```
xgb_cv = XGBClassifier(learning_rate=0.05,colsample_bytree= 0.7,max_depth=6,min_child_weight=11,missing=-999,
                       n_estimators= 1000,objective= 'binary:logistic',seed= 1337,
                       subsample= 0.8,silent=1, nthread=4)
```

```
accuracy = cross_val_score(xgb_cv,train_x,train_y, scoring='accuracy', cv = 10)
print(accuracy)
print("Accuracy of Model with Cross Validation is:",accuracy.mean() * 100)
```

```
[0.94788788 0.94196605 0.95380971 0.94630872 0.94352291 0.9462663
 0.95100751 0.94033979 0.9474516  0.94350059]
Accuracy of Model with Cross Validation is: 94.6206105585288
```

```
cv_xg = 94.6206105585288
print(cv_xg)
```

```
94.6206105585288
```

11.5 AdaBoost Classifier:

Boosting algorithms are a set of the low accurate classifier to create a highly accurate classifier. Low accuracy classifier (or weak classifier) offers the accuracy better than the flipping of a coin. Highly accurate classifier(or strong classifier) offer error rate close to 0. Boosting algorithm can track the model who failed the accurate prediction. Boosting algorithms are less affected by the overfitting problem.

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and

training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. Adaboost should meet two conditions:

1. The classifier should be trained interactively on various weighed training examples.
2. In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

Adaboost Prediction:

```
ada = AdaBoostClassifier()
ada.fit(train_x,train_y)
ada_pred = ada.predict(test_x)
```

```
score_ab_basic=accuracy_score(test_y,ada_pred)
print("accuracy:   %0.3f" % score_ab_basic)
from sklearn.metrics import classification_report
print(classification_report(test_y,ada_pred))
```

```
accuracy:   0.945
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      5974
           1       0.53      0.29      0.37       356

   micro avg       0.95      0.95      0.95      6330
   macro avg       0.74      0.64      0.67      6330
weighted avg       0.93      0.95      0.94      6330
```

Ada Boost Cross Validation:

```
ada = AdaBoostClassifier(random_state = 100,algorithm='SAMME.R', base_estimator=None,
                         learning_rate=1.0, n_estimators=50)
accuracy = cross_val_score(ada,train_x,train_y, scoring='accuracy', cv = 10)
print(accuracy)
print("Accuracy of Model with Cross Validation is:",accuracy.mean() * 100)
```

```
[0.94907225 0.93880774 0.93801816 0.9435452  0.94549763 0.9486369
 0.9470565  0.93717898 0.94310549 0.9454761 ]
Accuracy of Model with Cross Validation is: 94.36394948903221
```

```
cv_ab = accuracy.mean() * 100
cv_ab
```

```
94.36394948903221
```

11.6 Final Synopsis of the Models Applied:

	Accuracy Basic	Accuracy Final	CROSS VALIDATION	F1-Score Basic	F1-Score_Final	Precision Basic	Precision Final	Recall Basic	Recall Final
Random Forest	0.944866	0.946130	86.917753	0.350093	0.352941	0.519337	0.261236	0.264045	0.543860
Decision Tree	0.944866	0.945656	78.026301	0.350093	0.402778	0.519337	0.325843	0.264045	0.527273
XG-BOOST	0.949763	0.975678	94.620611	0.436170	0.975761	0.591346	0.972477	0.345506	0.979067
AdaBoost	0.945340	0.947393	94.363949	0.370909	0.915431	0.525773	0.917220	0.286517	0.913650

12.CONCLUSION

Summary of the Project Outcome

- The analysis can be divided into two parts, one serves the purpose of exploratory analysis and the other presents several classification models.
- We had explored many data handling techniques
- We had explored the smote technique for handling Imbalance problems
- We had explored the techniques, which can select the relevant features which when applied can give better results
- We had done cross validations which can show that the model is over fitting
- We can conclude that the problem with imbalanced class can be handled by boosting algorithms greatly.

13. REFERENCES

- https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63
- https://www.graphpad.com/support/faqid/1790/
- https://www.geeksforgeeks.org/exploring-correlation-in-python/
- https://www.datacamp.com/community/tutorials/adaboost-classifier-python
- https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb
- https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e
- https://elitedatascience.com/imbalanced-classes
- https://hub.packtpub.com/4-ways-implement-feature-selection-python-machine-learning/