Group Project Report

Perspectives on Business Analytics

Prof. Dr. Gruetzemacher

Barton School of Business

Credit Card Approval Prediction

Based on credit card data from Kaggle

**Contents**

1. **Introduction:**
   1. Problem Statement……………………………………………….3
   2. Data………………………………………………………………3
2. **Methodology:**
   1. Pre-Processing…………………………………………………………5
      1. Missing Values analysis………………………………………...7
      2. Outliers Analysis………………………………………….........7
      3. Exploratory Data Analysis……………………………………..9
      4. Feature Selection………………………………………………10

2.1.5 Feature Scaling…………………………………………………11

* 1. Modelling………………………………………………………………13
     1. Logistic Regression using Statmodels…………………………..13
     2. Cross Validation………………………………………………...14
     3. Random Forest….…………………………………………..15

1. **Results:**
   1. Model Evaluation………………………………………………….16
      1. Confusion Matrix……………………………………………16
      2. Accuracy…………………………………………………….16
      3. Recall………………………………………………………...17
      4. Precision……………………………………………………...17
   2. Model Selection……………………………………………………..18

**Appendix A -** Python Code…………………………………………..19

**References……………………………………………………………….** 21

## **INTRODUCTION**

**Background –**

Credit score cards are a common risk control method in the financial industry. It uses personal information and data submitted by credit card applicants to predict the probability of future defaults and credit card borrowings. The bank is able to decide whether to issue a credit card to the applicant. Credit scores can objectively quantify the magnitude of risk.

Generally speaking, credit score cards are based on historical data. Once encountering large economic fluctuations. Past models may lose their original predictive power. Logistic model is a common method for credit scoring. Because Logistic is suitable for binary classification tasks and can calculate the coefficients of each feature. In order to facilitate understanding and operation, the score card will multiply the logistic regression coefficient by a certain value (such as 100) and round it.

At present, with the development of machine learning algorithms. More predictive methods such as Boosting, Random Forest, and Support Vector Machines have been introduced into credit card scoring. However, these methods often do not have good transparency. It may be difficult to provide customers and regulators with a reason for rejection or acceptance.

**1.1 Problem Statement**

This dataset's objective is predicting whether an applicant's credit card approval will be approved or not. The dataset contains applicants' basic information and applicants' credit history. There are **433887** rows in application.csv. ID is from **5008804** to **7999952**. In credit\_record.csv, there are **1048575** rows of **45985** ID's credit record. ID is from **5001711** to **5150487**.

**1.2 Data**

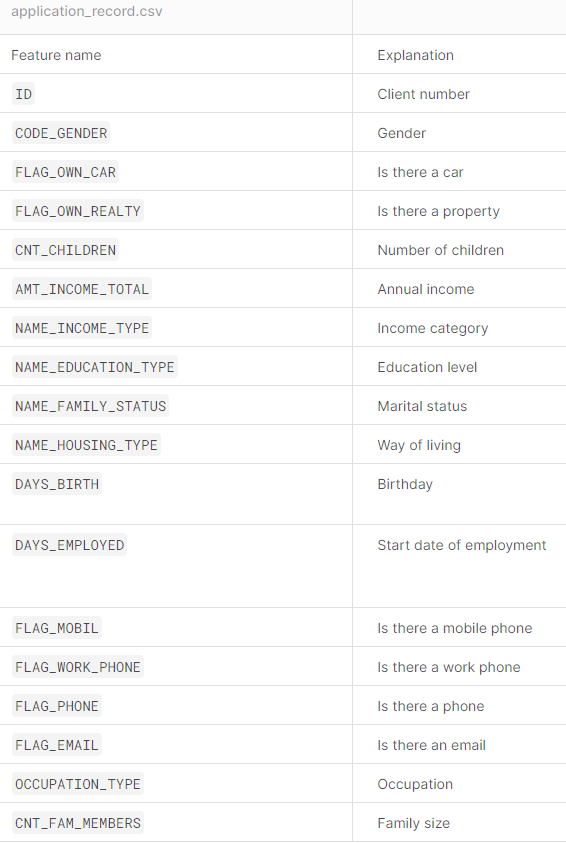
The task is to develop a classification model that will classify the customers into potential ‘good’ or ‘bad’ client.

There are two tables, which are connected by ID.

application\_record.csv contains appliers personal information, which you could use as features for predicting.

credit\_record.csv records users’ behaviours of credit card.

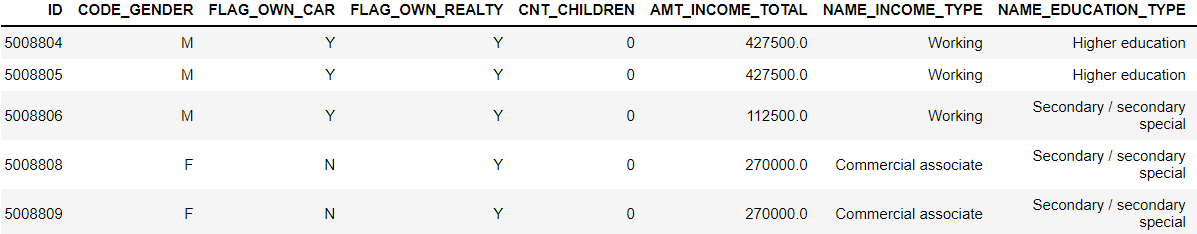
**application\_record.csv attributes:**

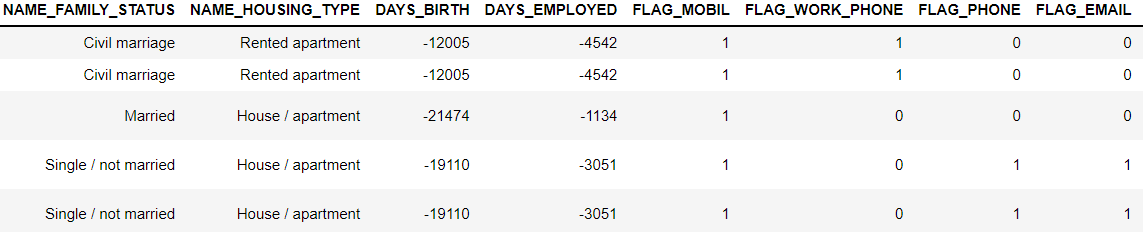


DAYS\_BIRTH feature is represented as count backwards from current day (0), -1 means yesterday.

DAYS\_EMPLOYED feature is represented as count backwards from current day(0). If positive, it means the person currently unemployed.

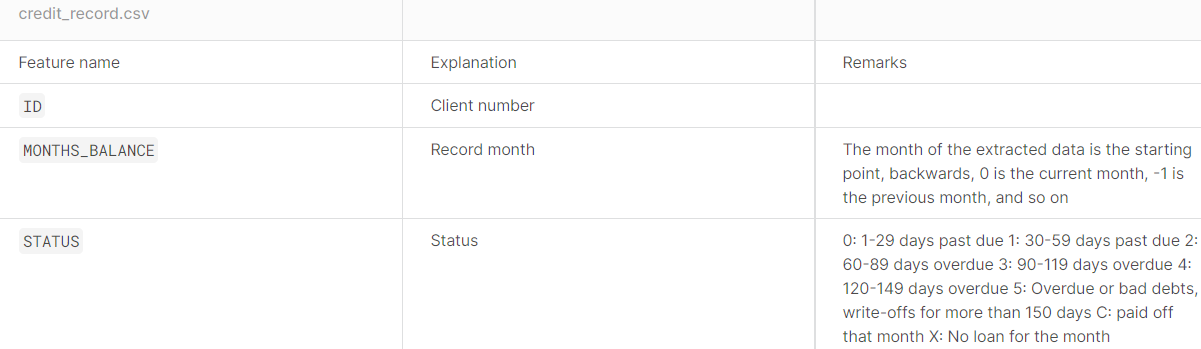
Here’s how application\_record.csv data looks like:



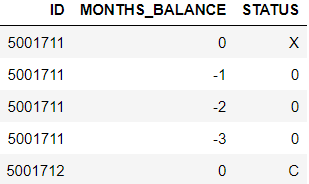


application data has 438557 rows and 18 columns.

**credit\_record.csv attributes:**



credit\_record.csv data looks like:



with 1048575 rows and 3 columns.

Here, the STATUS feature is very important as it is the target variable for the model. As credit\_record.csv is the users behaviour of credit card. The STATUS is classified into eight categories.

0: 1-29 days past due 4: 120-149 days overdue

1: 30-59 days past due 5: Overdue or bad debts, write-offs for more than 150 days

2: 60-89 days overdue C: paid off that month

3: 90-119 days overdue X: No loan for the month

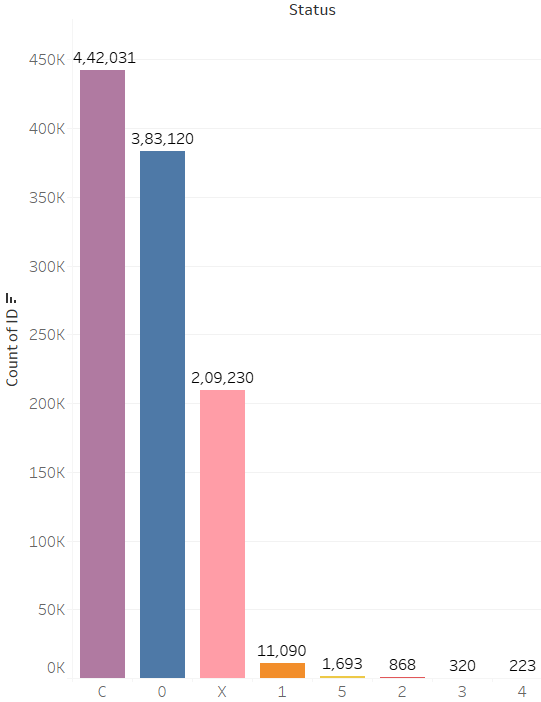


Fig. 2.1 bar-graph for distribution in STATUS feature

categorizing 'STATUS' column to binary classification

[C, X, 0] =>0: Good Client and [1,2,3,4,5] => 1: bad client

Both the datasets can be merged on ‘ID’ 36457 records can be matched.



After merging both the datasets we have,

36457 rows, 19 columns

# **2.METHODOLOGY**

**2.1** **Pre-Processing**

Any classification model requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data. Now, that both the datasets are merged we have some missing values in the final data.

**2.1.1 Missing Values Analysis:**

In classification modelling we have to look into data weather the data is clean, does it have any missing values. If it does have any missing then those values have to be replaced by mathematical methods or if the missing data number is insignificant, they can be removed.

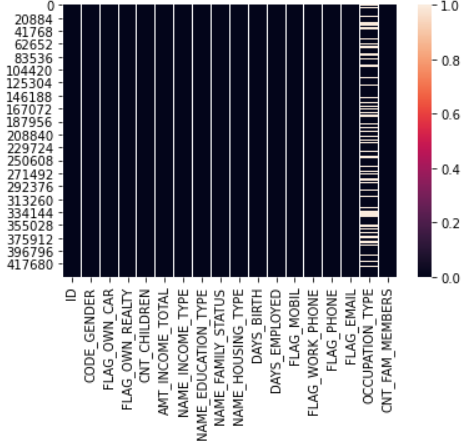


Fig. 2.2 heat-map for null-values in features

From the above visualization it is clear that occupation\_type, has missing values as it is a categorical variable 11323 rows has missing values from 36457 rows which is around 30% as it is a categorical variable. we are going to remove them. After removing we are left with 25134 rows.

**2.1.2 Outlier Analysis:**

An outlier is a value or point that [differs substantially from the rest of the data](https://en.wikipedia.org/wiki/Outlier). Outliers in the data may occur due to poor measurement quality or some external reasons. As they may affect in our prediction modelling, we have to deal with them. In a simple way we can detect outliers by plotting box plots of the different variables in the data set.

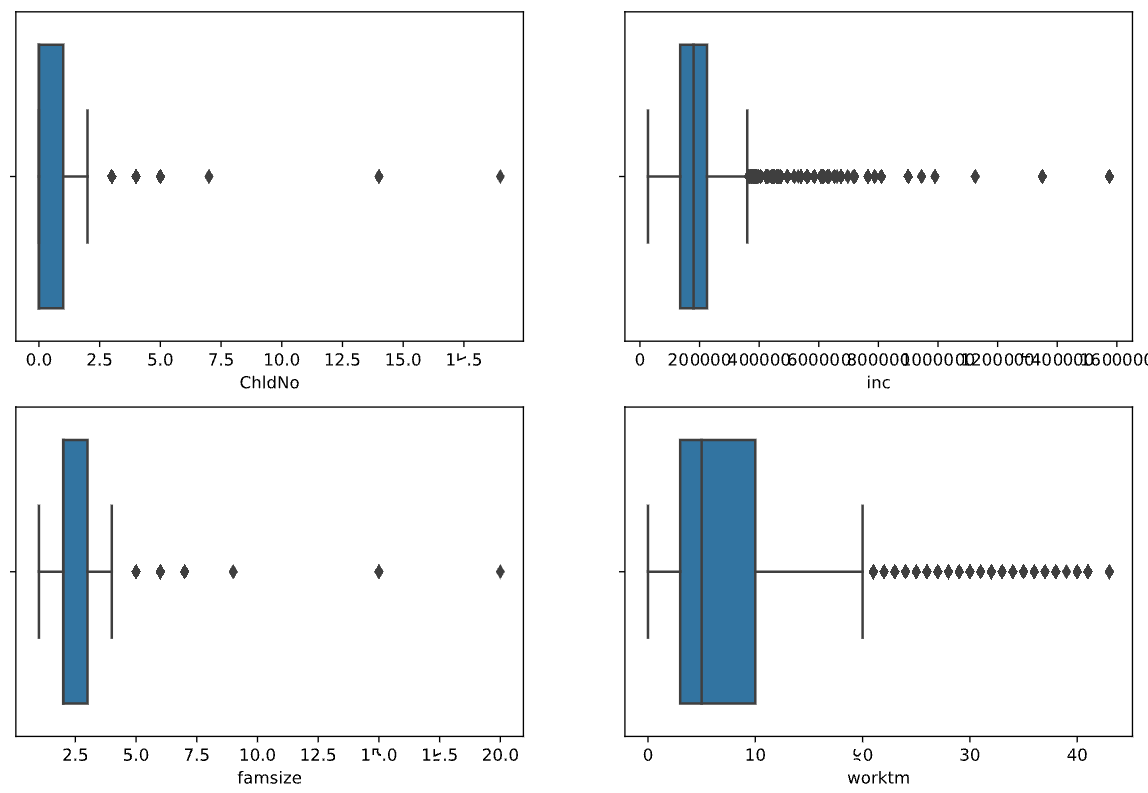


Fig. 2.3 box-plots for outliers in features

An outlier is a data point that differs significantly from other observations. Box plot diagram is a graphical method typically depicted by quartiles and inter quartiles that helps in defining the upper limit and lower limit beyond which any data lying will be considered as outliers.

From this we can see that ChldNo, inc, famsize, worktm columns have outliers so this may affect the model so we have to detect them as we have thousands of observations, we can remove them.

Here is the method to remove the outliers from the data with the 1st and 99th percentile calculating the MIN and MAX then removing all the data points less than MIN and greater than MAX.

After removing all the outliers from the data-set we are left with 25043 rows.

**2.1.3 Exploratory Data Analysis:**

Organizing, Plotting and Summarizing the data.

By visualising the data, we can know how the data is distributed and then we can detect the pattern in the data.

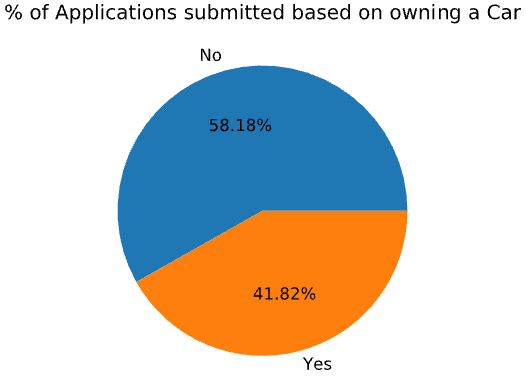
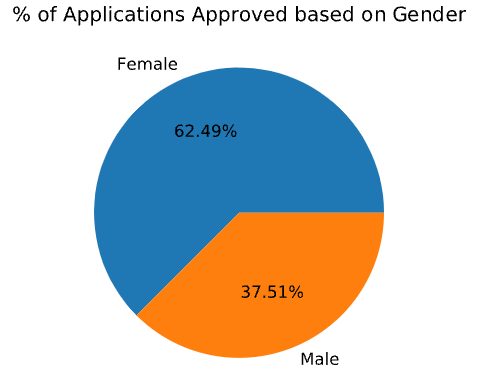


Fig. 2.4 pie-charts of gender dist. & car ownership

From the visualizations it is clear that most of the applications submitted and approvals are females. It is interesting to know that majority of applications do not own a car.

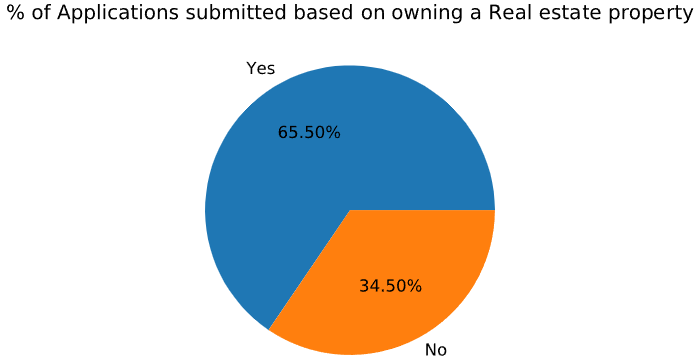


Fig. 2.5 pie-chart of Real-estate property ownership

Majority of the applications own real-estate property

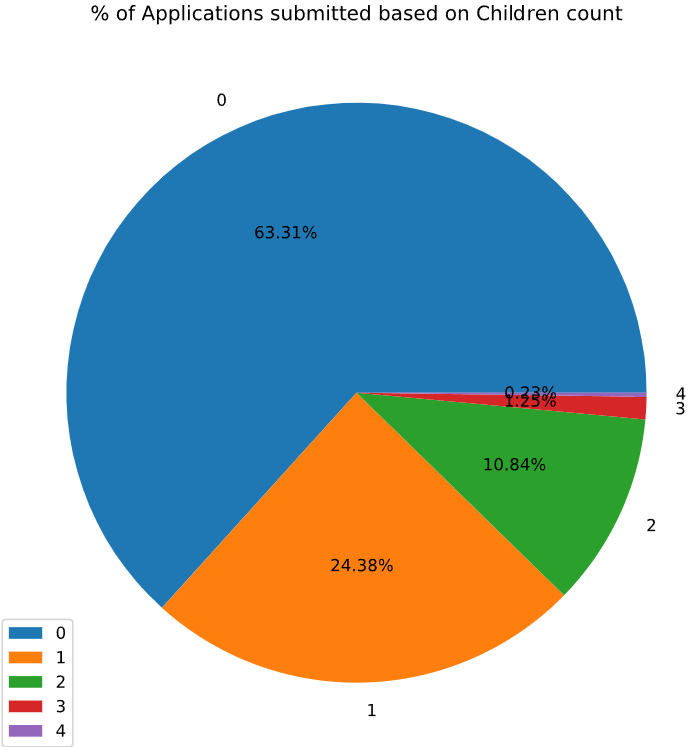
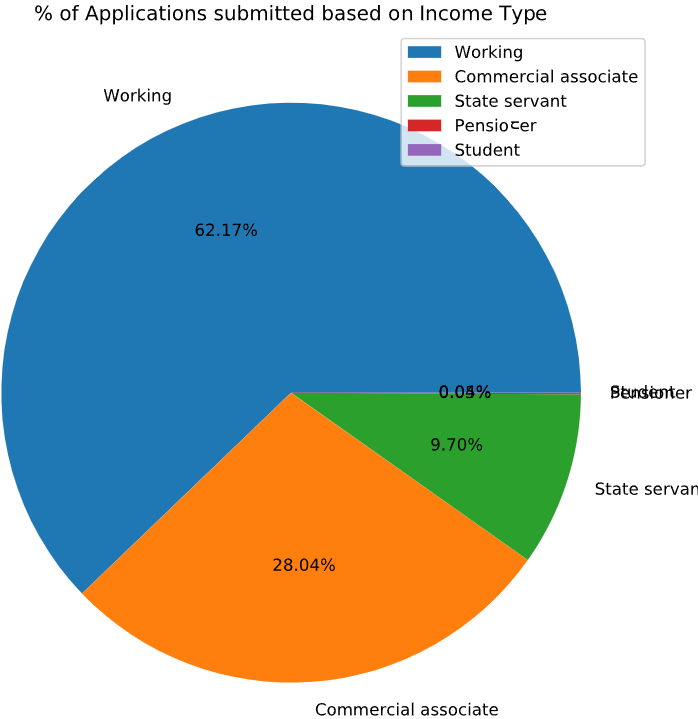
 

Fig. 2.6 Pie-charts for categorical dist. of children & Income-type

Majority of the applicants don’t have children and the distribution of the income type most of them are working type.

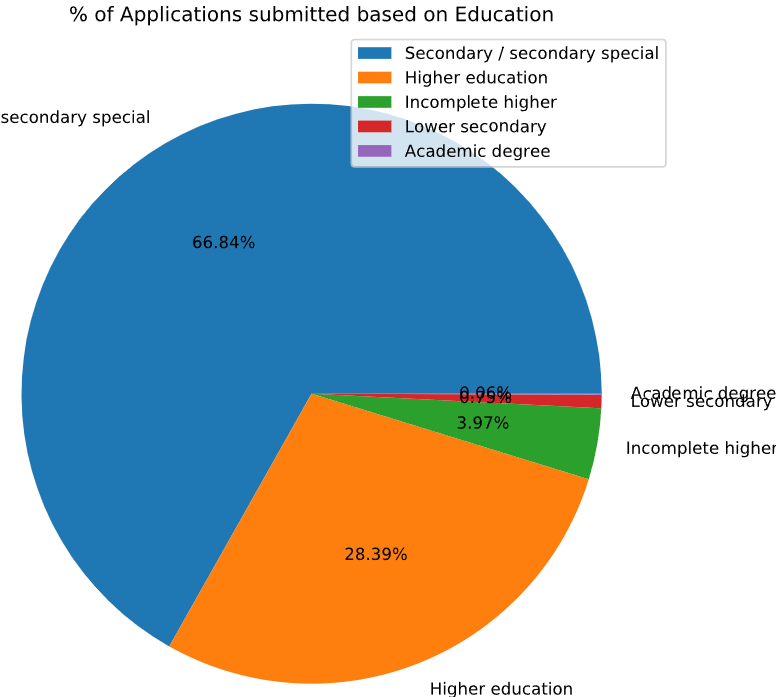
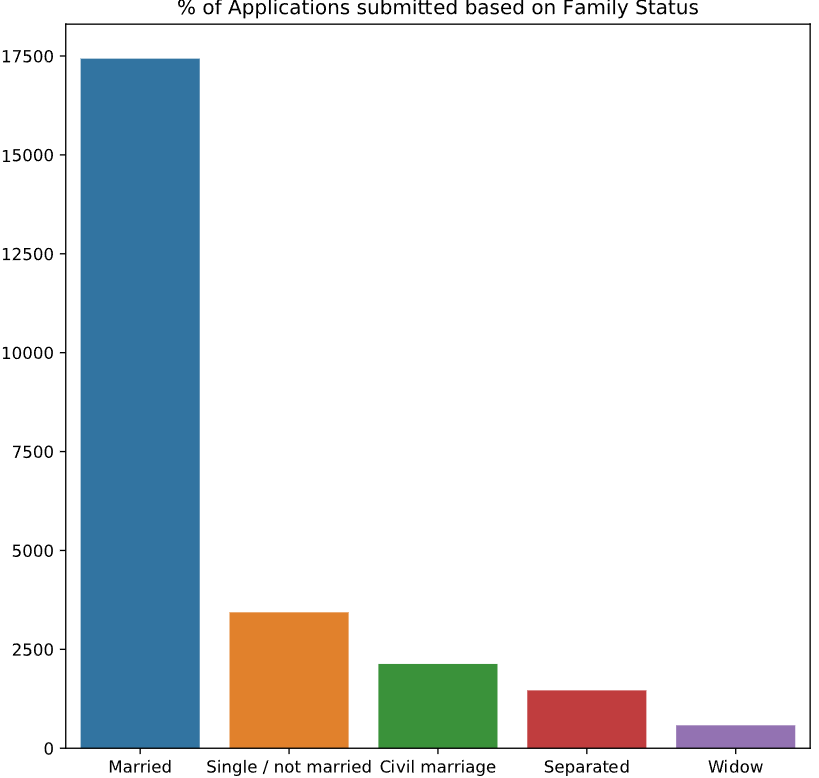
 

Fig. 2.7 Pie-chart of applicants Education Fig. 2.8 Bar-chart of family status

from the above plots we can see that most of the applicants are secondary education and major of them are married.

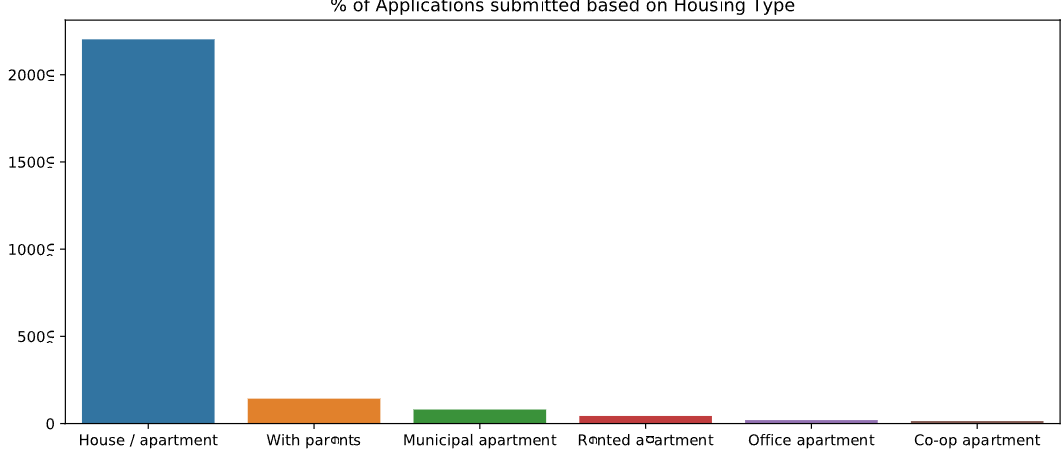


Fig. 2.9 Bar-chart of family status

As the Age is given in terms of days, we have to convert them to years in the same way work-experience is in terms of days so converting it into years as well.

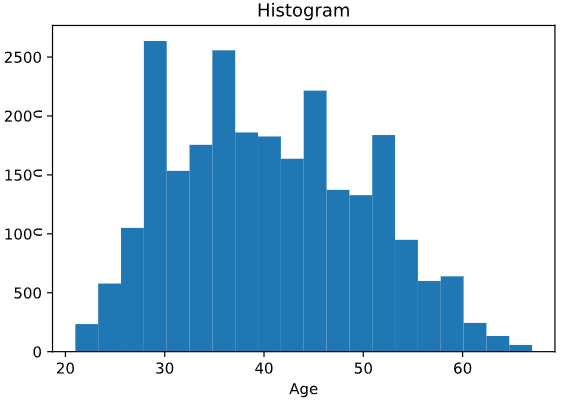
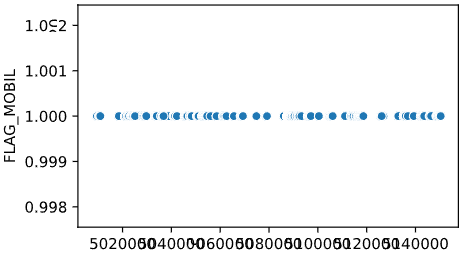


Fig. 2.10 Histogram for Age distribution



Flag\_Phone column has only one value through out the data set so removing it from the data.

**2.1.4 Feature Selection:**

In the classification modelling feature selection is about selecting the independent variables which will be helpful in predicting the target variable. It is also know as Dimensionality Reduction. For numerical data we can use correlation plot

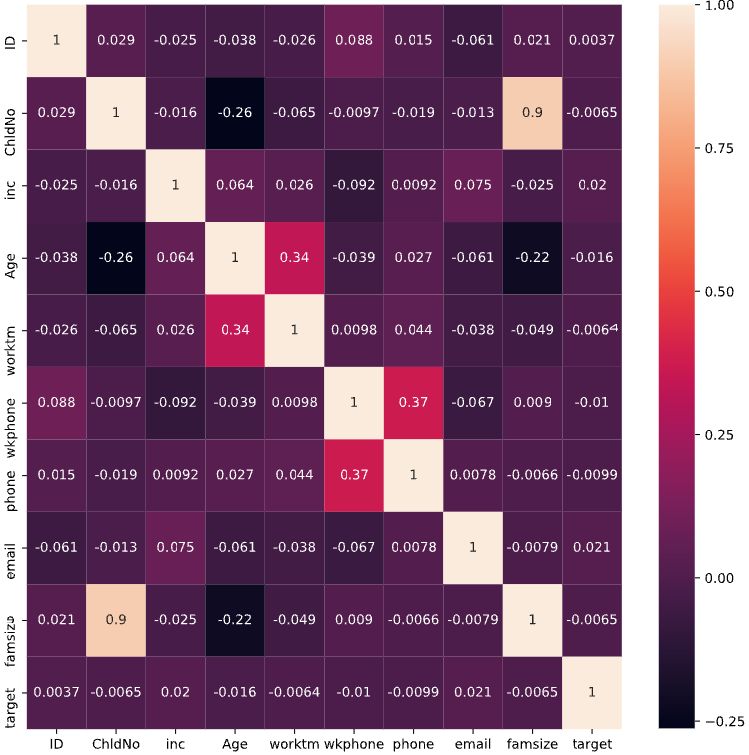


Fig. 2.4

Correlation:

1. It show whether and how strongly pairs of variables are related to each other.

2. Correlation takes values between -1 to +1, wherein values close to +1 represents strong positive correlation and values close to -1 represents strong negative correlation.

3. Value 0 represents no relation between variables. Since all the independent variables are of numeric type no need to perform chi square test as it is only applicable if there is any categorical variable in the dataset.

Here, ChldNo and famsize are highly correlated so any one of the feature can be used in the analysis.

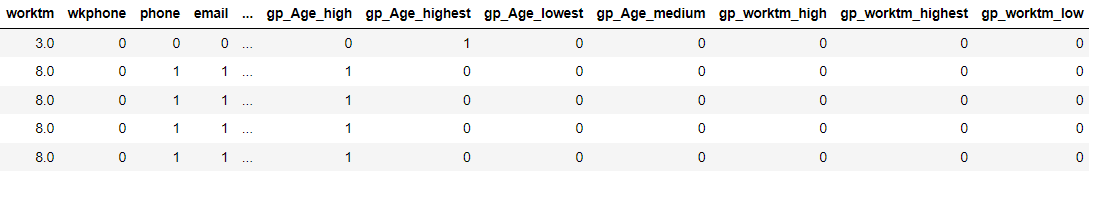
**2.1.5 Feature Scaling:**

While working to learn about data it is important to scale the features to a range which are centered around zero. It is done to bring the feature on a same scale. This is done so that the variance of the features are in the same range. If a feature’s variance is orders of magnitude more than the variance of other features, that particular feature might dominate other features in the dataset, which is not something we want happening in our model. Before choosing the data scaling method, we need to check the distribution of data. If the data is uniformly distributed, then Standardization is the suitable method for the scaling purpose. On the other hand, if the data is not normally distributed, we go with Normalization scaling method. Normalization is bringing all the variables into proportion with one another. Normalization is the process of reducing unwanted variation either within or between variables and the range lies between 0 and 1. But here the data we are dealing with has various types of data like categorical, continuous and numerical data.

Categorical variables are those which can take values from a finite set. For such variables, the values that they take up can have an intrinsic ordering (for e.g., speed: {low, medium, high}). Such variables are called as Ordinal Categorical Variables. On the other hand, some categorical variables may not have any intrinsic ordering (for e.g., Gender: {Male, Female}). Such categorical variables are called as Nominal Categorical Variables.

The continuous variables we have like Age, work experience, income are converted into categories by dividing them into bins and we are going to convert categorical variables columns to numerical as a part of one-hot encoding and we have categorical variables and we can convert them to dummy variables so that the dataset is ready for modelling.

pd.get\_dummies ([documentation](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html)) returns a new data-frame that contains one-hot-encoded columns. We can observe that not all the columns were encoded. This is because, if no columns are passed to pd.get\_dummies so as to tell which columns to one-hot-encode, by default it takes the columns with data-type ‘object’. It then encodes them and returns a new data-frame with new columns that replace the old categorical column.



As we have observed that the data set which is provided to us is an imbalanced data set the target variables in the target column are not equally distributed. So, we try resampling the target variable as we have 88% of 0’s and only 12% of 1’s .

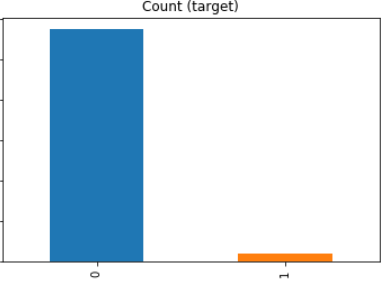
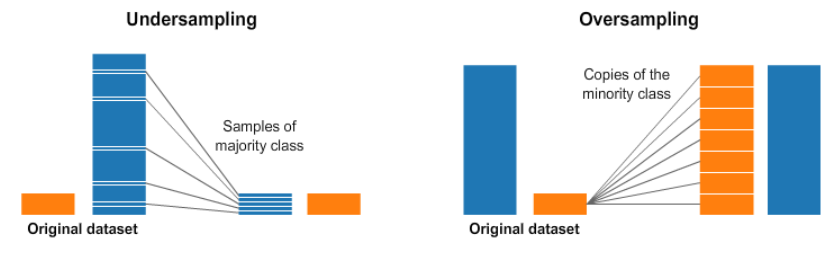


Fig. 2.x

we use over sampling technique so that 1’s match with the 0’s as well so that the model cannot be baised.

**Resampling:**

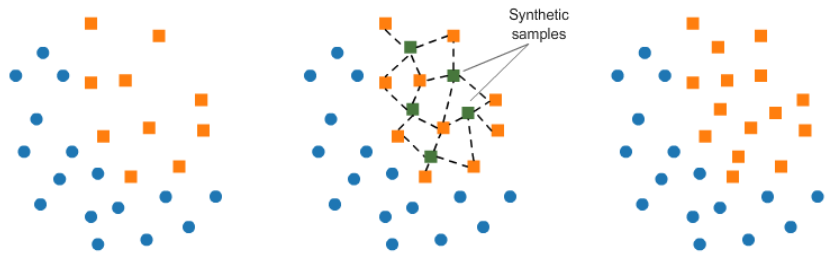
A widely adopted technique for dealing with highly unbalanced datasets is called resampling. It consists of removing samples from the majority class (under-sampling) and / or adding more examples from the minority class (over-sampling).



Despite the advantage of balancing classes, these techniques also have their weaknesses. The simplest implementation of over-sampling is to duplicate random records from the minority class, which can cause overfitting. In under-sampling, the simplest technique involves removing random records from the majority class, which can cause loss of information.

## **Over-sampling: SMOTE**

SMOTE (Synthetic Minority Oversampling Technique) consists of synthesizing elements for the minority class, based on those that already exist. It works randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.



After sampling we have a balanced data set,

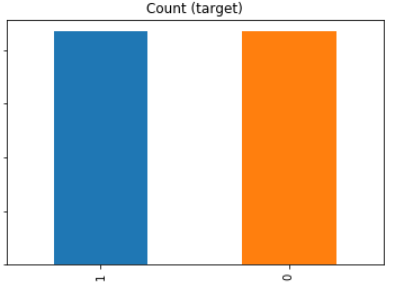
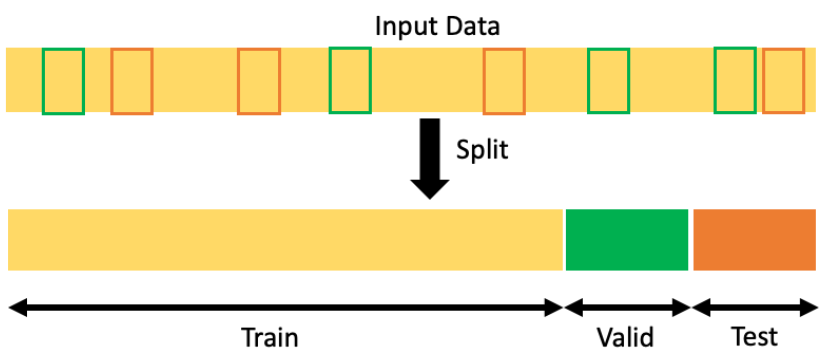


Fig. 2.5

**2.2 Modelling:**

In modelling we first have to split the clean dataset to train-set and test-set and then develop different models and evaluate them by metrics.



**You can split the dataset into train and test set using the [train\_test\_split()](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html" \t "_blank) method of the sklearn library.**

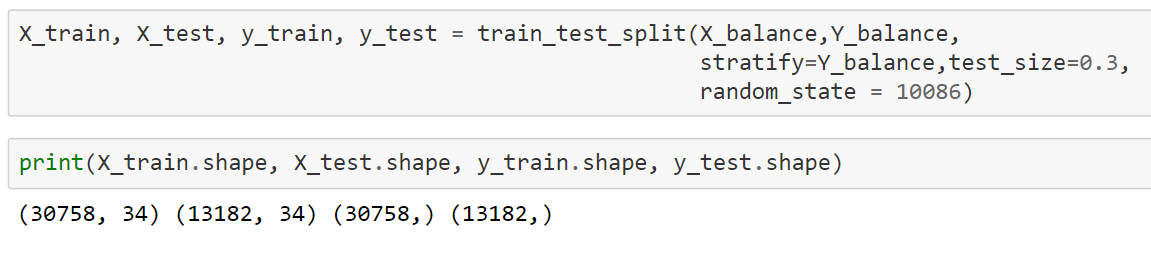
It accepts one **mandatory parameter.**

* Input Dataset – It is a sequence of array-like objects of the same size. Allowed inputs are lists, NumPy arrays, scipy-sparse matrices, or pandas data frames.

It also accepts few other **optional parameters**.

* test\_size – Size of the test dataset split. It normally accepts float or int type of values. If you want to have 25% of the data for testing, you can pass 0.25 as test\_size = 0.25. If it is set to None, the size will be automatically set to complement the train size. If the Train\_size is also None, then it’ll be set to 0.25.
* train\_size – Size of the train dataset split. It normally accepts float or int type of values. If you want to have 75% of the data for training, you can pass 0.75 as train\_size = 0.75. If it is set to None, the size will be automatically set to complement the test size. If the test\_size is also None, then it’ll be set to 0.75.
* random\_state – It is an int type parameter. It controls the shuffling applied to the dataset before splitting it into two sets.
* shuffle – It is a boolean type parameter. It is used to denote whether shuffling must be done before the split. If shuffling is False, then the next parameter, stratify must be None.
* stratify – array-like object. It is used to split the data in a stratified fashion using the class labels.

Snippet:



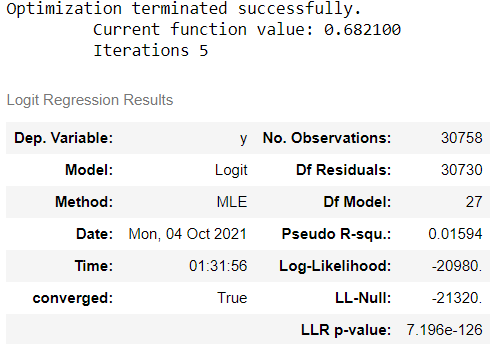
**2.2.1 Logistic Regression using Statsmodels**

**Logistic regression** is the type of regression analysis used to find the probability of a certain event occurring. It is the best suited type of regression for cases where we have a categorical dependent variable which can take only discrete values.

**Statsmodels** is a Python module that provides various functions for estimating different statistical models and performing statistical tests

First, we define the set of dependent(**y**) and independent(**X**) variables. If the dependent variable is in non-numeric form, it is first converted to numeric using dummies. The file used in the example for training the model, can be downloaded [here](https://drive.google.com/file/d/1g4Ib_zuG_hJG6VWlXvqANti69w5cM66K/view?usp=sharing).

Statsmodels provides a **Logit()** function for performing logistic regression. The *Logit()* function accepts **y** and **X** as parameters and returns the *Logit* object. The model is then fitted to the data.



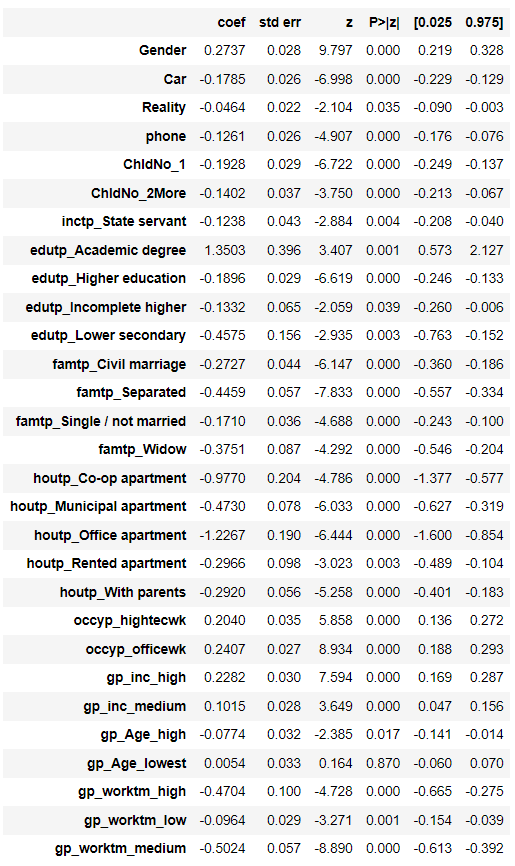
Explanation of some of the terms in the summary table:

**coef :** the coefficients of the independent variables in the regression equation.

**Log-Likelihood :** the natural logarithm of the Maximum Likelihood Estimation(MLE) function. MLE is the optimization process of finding the set of parameters that result in the best fit.

**LL-Null :** the value of log-likelihood of the model when no independent variable is included(only an intercept is included).

**Pseudo R-squ. :** a substitute for the [R-squared](https://www.geeksforgeeks.org/ml-r-squared-in-regression-analysis/) value in Least Squares linear regression. It is the ratio of the log-likelihood of the null model to that of the full model.



The p-values of all the variables are less than 0.05 except one, hence statistically significant.

## **3.RESULTS**

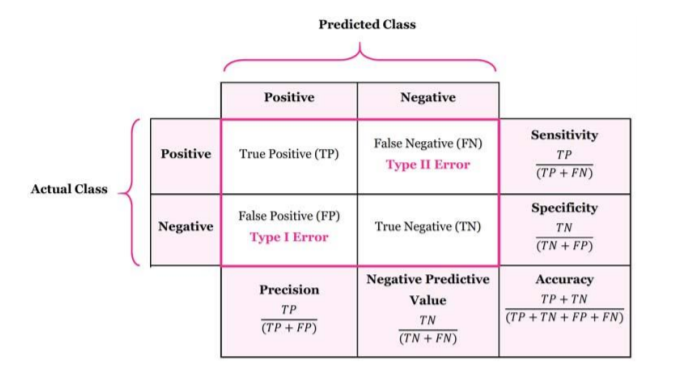
**Model Evaluation:**

For Evaluating a classification model we have regression metrics they are:-

* 1. **Logistic Regression Metrics:**

**3.1.1 Confusion Matrix**

In machine learning, confusion matrix is one of the easiest ways to summarize the performance of your algorithm. At times, it is difficult to judge the accuracy of a model by just looking at the accuracy because of problems like unequal distribution. So, a better way to check how good your model is, is to use a confusion matrix.



**3.1.2 Accuracy**

It is the most intuitive performance measure and it simply a ratio of correctly predicted to the total observations.

**3.1.3 Sensitivity / True Positive Rate / Recall**

Sensitivity formula

Sensitivity tells us what proportion of the positive class got correctly classified.

**3.1.4 False Negative Rate**

False Negative Rate

False Negative Rate (FNR) tells us what proportion of the positive class got incorrectly classified by the classifier.

A higher TPR and a lower FNR is desirable since we want to correctly classify the positive class.

### **3.1.5 Specificity / True Negative Rate**

Specificity formula

Specificity tells us what proportion of the negative class got correctly classified.

**3.1.6 False Positive Rate**

False Positive Rate

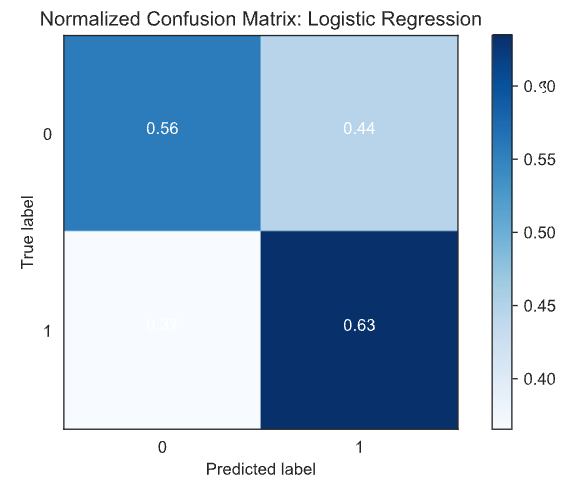
FPR tells us what proportion of the negative class got incorrectly classified by the classifier.

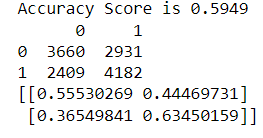
A higher TNR and a lower FPR is desirable since we want to correctly classify the negative class.

**3.1.4 Precision**

It is also called as the positive predictive value. Number of correct positives in your model that predicts compared to the total number of positives it predicts.

Precision = True Positives / (True Positives + False Positives)





The result is telling us that we have **3660+4182** correct predictions and **2409+2931** incorrect predictions.

**Compute precision, recall, F-measure and support:**

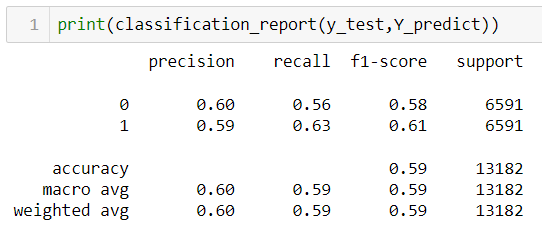
The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier to not label a sample as positive if it is negative.

The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

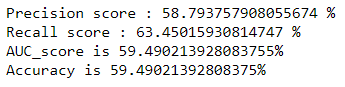
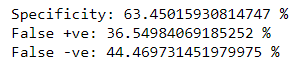
The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

The F-beta score weights the recall more than the precision by a factor of beta. beta = 1.0 means recall and precision are equally important.

The support is the number of occurrences of each class in y\_test.



**Interpretation**: Of the entire test set, 59% of the credit card applications were predicted as approved of the entire test set, approved applications.

**Sensitivity / recall** tells us what proportion of the positive class got correctly classified from the above results 59% of approvals are classified as approvals.

**Specificity** tells us what proportion of the negative class got correctly classified. So, we have 63% of rejections are classified as rejections.

**Precision** is number of positive approvals the model classified compared to the total number of approvals it classified. We have a precision of 58% which means out of all the approvals it classified 58% of them are right.

**False Positive** tells us what proportion of the reject class got incorrectly classified by the classifier from the results 36% of rejections are classified as approval by the model.

**False Negative** tells us what proportion of the approval class got incorrectly classified by the classifier. So, we have 44% of approval class is classified as rejections.

A higher TPR(59%) and a lower FNR(42%) is desirable since we want to correctly classify the approval class.

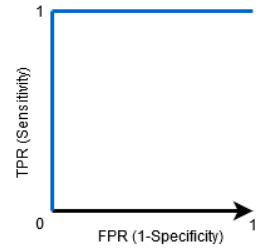
A higher TNR(63%) and a lower FPR(36%) is desirable since we want to correctly classify the rejection class.

## **4.RESULTS-DISCUSSION**

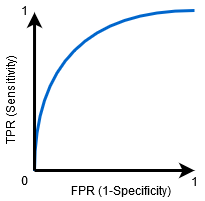
**4.1 AUC-ROC Curve:**

The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR**against **FPR**at various threshold values and essentially **separates the ‘signal’ from the ‘noise’**. The **Area Under the Curve (AUC)**is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

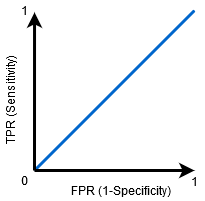
The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.

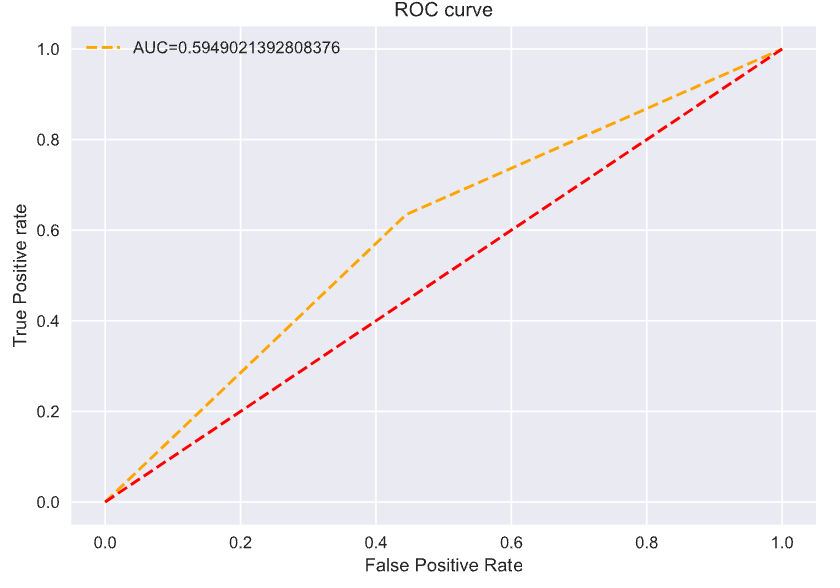


When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.



When AUC=0.5, then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting random class or constant class for all the data points.

So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes.



The ROC curve is a common tool used with binary classifiers. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

**4.1 K-Fold Cross-Validation:**

It tries to address the problem of the holdout method. It ensures that the score of our model does not depend on the way we select our train and test subsets. In this approach, we divide the data set into k number of subsets and the holdout method is repeated k number of times.

**Steps:**

Randomly divide the entire dataset into k numbers of folds.

For each fold the dataset, build your model on k – 1 folds of the dataset and test the model to find the performance for the kth fold.

Repeat this until each of the k-folds has become the test set exactly once.

Finally, the average of k -accuracies is called the cross-validation accuracy and it will serve as our performance metric for the model.

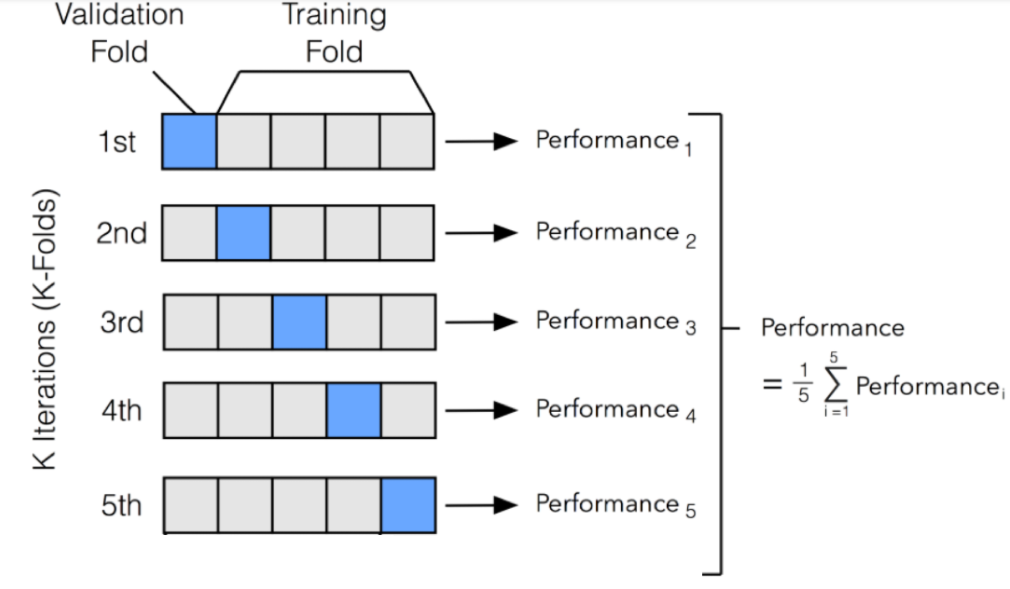


Fig. 4.1

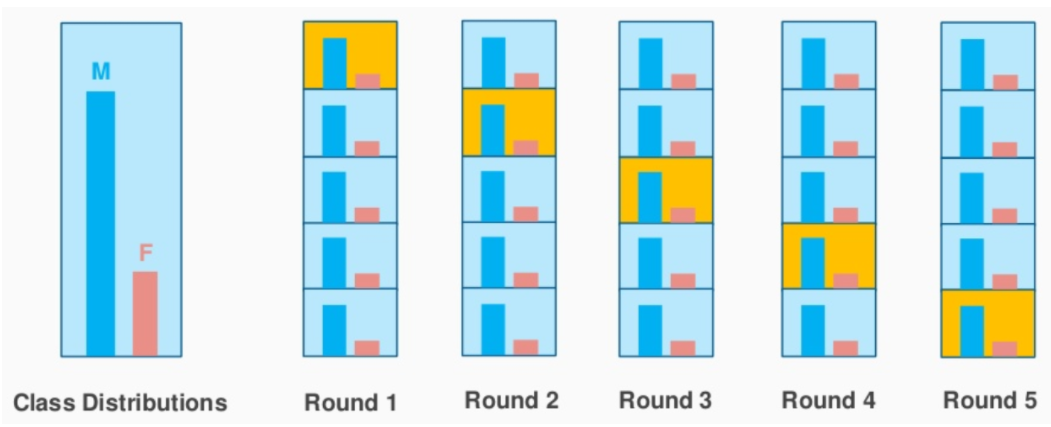
**4.3 Stratified K-Fold Cross Validation:**

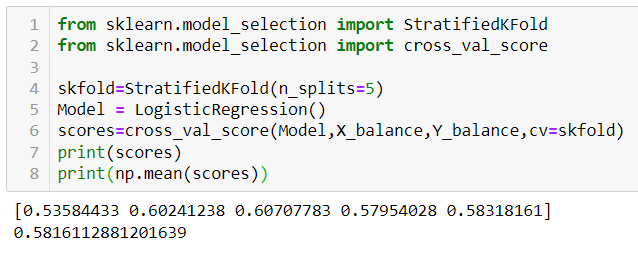
It tries to address the problem of the K-Fold approach.

Since, In our previous approach, we first randomly shuffled the data and then divided it into folds, in some cases there is a chance that we may get highly imbalanced folds which may cause our model to be biased towards a particular class.

**For example,** let us somehow get a fold that contains the majority of the samples from the positive class and only a few samples from the negative class. This will certainly affect our training and to avoid this we make the stratified folds using the **stratification**process.

**Stratification** It is the process of rearranging the data such that each of the folds is a good representative of the whole dataset w.r.t different classes.





We have developed a classification model for predicting the approval or rejection of credit card application from the results it is clear that the model accuracy is 58%. Thought the accuracy for the model isn’t that great but it’s still doing a pretty good job. As we know the credit record dataset is a multi-categorised for our convience we have converted it into binary classification and the data predictive power declined as we have to deal with the imbalanced dataset using resampling techniques.

As we have a higher True Positivity rate and a lower False Negative Rate it means that we have correctly classified the approval class and a higher True Negative rate and a lower False Positive Rate indicates that model is classifying majority of rejection class as well.

5.CONCLUSION

Despite the limitations discussed in the previous chapter, it can be stated that the developed model has a strong classification quality for the underlying data. Moreover, most of the features included have a strong statistical significance and all the assumptions concerning the logistic regression are fulfilled. However, the data itself is imbalanced at the target level. Although the imbalance in the dataset is dealt with resampling techniques but still it has its own draw-backs. The predict power of the original data declines as the resampling technique removes some record of a class and adds the records of the minority class by performing k-nearest neighbour.

Furthermore, from the summary table we can see that in the classification model features like gender, Education, family type, house type has high coefficient comparative to other features. Even from observing the p-value we know that all the features that are giving to the model are statistically significant. So, the developed model has a good classification quality.

As from the actual data we can perform a multi-category classification and classify applicant as per there risk profile but it will exceed the scope of this class. We choose to categorize the dependent variable and use a binary classification model for this project.

**REFERENCES:**

* <https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/>
* [**https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/?#**](https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/?#)
* <https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>
* <https://www.kaggle.com/rikdifos/credit-card-approval-prediction>
* <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html>
* <https://medium.com/ibm-data-science-experience/markdown-for-jupyter-notebooks-cheatsheet-386c05aeebed>