APS Failure at Scania Trucks Data Set

EE 660 Course Project

Project Type: (1) Design a system based on real-world data;

Chaitra Suresh

csuresh@usc.edu

12/09/2019

1. Abstract

I have attempted this project individually and have worked on the "APS Failure at Scania Trucks Data Set". The dataset consists of positive class (failure due to specific components of APS – Air Pressurized System) and negative class (failure due to other components not related to APS). The problem statement is to determine/minimize the total cost prediction model by correctly identifying the cases where the failure is due to APS.

The APS Failure at Scania Trucks data is associated with the classification task. There are in total 170 features with numeric attribute characteristics and 60,000 instances including missing values. Due to computation time, 'smaller' dataset with 19,999 training and 16,000 testing instances are considered for all the evaluations and analysis.

The total cost prediction of a model is characterized by the sum of cost involved for unnecessary checks needed to be done by a mechanic and cost involved for missing a faulty truck, which may cause breakdown due to APS failure. The cost involved for missing a faulty truck is 500 while the cost involved for unnecessary check is 10. The best model should have less false negative and false positive samples, in order to minimize the total cost of prediction model.

This project involved the following steps of pre-processing the data by evaluating for the possible missing values (169 features had missing values amongst 170 features), compensation for unbalanced dataset, feature extraction, training and classification, choice of hyper parameters through cross validation.

Several classifiers like Random Forest Classifier, Logistic Regression, k-NearestNeighbour (KNN) was adopted for the classification problem. Each of them is associated with different pre-processing steps to evaluate the performance with attributes like accuracy, confusion matrix, F1 score and interpret the impact of preprocessing stages on the classifier.

The best test total cost obtained for Random Forest Classifier for training on balanced data set is 14060.

2. Introduction

2.1. Problem Type, Statement and Goals

- The main idea of the project is to minimize the total cost prediction of a model, by minimizing the false negative and false positive instances of a classification problem – identification of positive class (failure due to specific components of APS – Air Pressurized System) and negative class (failure due to other components not related to APS).
- Total cost = 500 * False negative instances + 10 * False positive instances
- The cost involved for missing a faulty truck (500) is higher compared to the cost involved for unnecessary check (10).
- The problem is interesting due to following reasons:
    - High dimensionality of feature space - 170 dimensions
    - 850015 missing values in 60000 training samples. (Smaller training set has 284543 missing values in 19999 training samples)
    - Highly imbalanced dataset with 1000 positive class and 59000 negative class in 60000 training samples. (Smaller training set with 333 positive class and 19666 negative class)
    - Significant amounts of preprocessing required for the above reasons.

2.2. Prior and Related Work - None

2.3. Overview of Our Approach

- The training set with 60000 training samples (positive class of 1000 and negative class of 59000 instances) takes more than 2 hours to model a classifier since using SMOTE, the training samples size increases to 118000.
- Therefore, a smaller training set with 333 positive class and 19999 negative class instances are used with a total of 19999 training samples. The smaller training set is the same dataset which was given during EE559 project.
- Features are the bins of histogram of a measurement. The measurement type, their conditions and the attribute names of the data have been anonymized for proprietary reasons. There is no distinguishing remark which can be made by analyzing each of 170 feature because of highly imbalanced samples for each class and missing values in 169 feature.
- Different classifiers with hyper parameter tuning are evaluated on total cost of the prediction model. Performance metrics like accuracy, F1 score, precision, recall are also considered.

3. Implementation

3.1. Data Set

- The original dataset consists of 60000 training samples with 1000 positive class and 59000 negative class instances.
- There are 170 features. Features are the bins of histogram of a measurement. The measurement type, their conditions and the **attribute names of the data have been anonymized for proprietary reasons**.
- The datatype of all the features are numeric. Below is a figure depicting the statistical description of few features. 169 features have missing values out of 170 features.

```
df.describe(include='all')
```

	class	aa_000	ab_000	ac_000	ad_000	ae_000	af_000	ag_000	ag_001	ag_002	ag_003	ag_004	ag_005	ag_006	ag_007	ag_008
count	60000.000000	6.000000e+04	13671	56665	45139	57500	57500	59329	59329	59329	59329	59329	59329	59329	59329	59329
unique	NaN	NaN	29	2061	1886	333	418	154	617	2422	7879	23071	40797	40615	32125	18932
top	NaN	NaN	0	0	0	0	0	0	0	0	0	0	0	0	0	0
freq	NaN	NaN	10977	8752	2009	55543	55476	59133	58587	56181	46894	1305	1941	1238	15875	25091
mean	-0.966667	5.933650e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	0.256040	1.454301e+05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	-1.000000	0.000000e+00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	-1.000000	8.340000e+02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	-1.000000	3.077600e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	-1.000000	4.866800e+04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	1.000000	2.746564e+06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

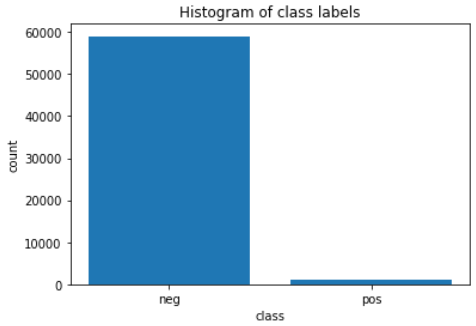
11 rows × 171 columns

- The original data set consists of 60000 training samples which takes more than 2 hours to model a classifier, in which training samples increases to 118000 samples using SMOTE (Synthetic Minority Over-sampling Technique). Below is a figure depicting few data samples from smaller training dataset with 333 positive class and 19666 negative class samples (same smaller_set given in EE559 project)

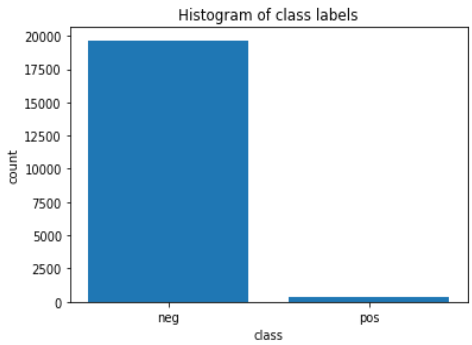
	class	aa_000	ab_000	ac_000	ad_000	ae_000	af_000	ag_000	ag_001	ag_002	ag_003	ag_004	ag_005	ag_006	ag_007	ag_008	ag_009
0	-1	46658	NaN	2962	NaN	0	0	0	0	0	0	38364	964660	1932028	390626	11746	0
1	-1	280	0	44	22	0	0	0	0	0	1404	2644	24800	24474	0	0	0
2	-1	8	NaN	6	6	0	0	0	0	0	0	0	0	2274	0	0	0
3	-1	224	0	48	42	0	0	0	0	0	0	2096	47562	31194	874	0	0
4	-1	31622	NaN	456	NaN	0	0	0	0	0	0	356	36990	1471402	827546	10094	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19994	1	569938	NaN	5022	NaN	0	0	0	0	220	327566	7479072	20942430	17554492	804236	60418	292
19995	1	741594	NaN	NaN	6896	NaN	NaN	0	59672	522892	8075256	24300822	24014152	8067500	1226602	36474	0
19996	1	312724	0	1702	1506	0	0	0	2462	312426	3453906	8969682	7908650	2165718	185852	10060	0
19997	1	718878	0	NaN	NaN	0	0	0	0	148180	1716386	6819930	4208700	998450	99194	2196	0
19998	1	643780	NaN	NaN	NaN	NaN	NaN	0	0	0	0	42284	4018570	27921852	13050722	324194	6708

19999 rows × 171 columns

- The original training set is highly imbalanced data set with 1000 positive and 59000 negative class samples



- The smaller training set is highly imbalanced data set with 333 positive and 19666 negative class samples



- The number of missing values in original training set with 60000 training samples is **850015**.
- The number of missing values in smaller training set with 19999 training samples is **284543**.
- The test data consists of 15625 negative class samples and 375 positive class samples.

3.2. Preprocessing, Feature Extraction, Dimensionality Adjustment

- Pre-processing of data is an important step towards machine learning/ pattern recognition in order to have a meaningful representation of the data in order to extract maximum patterns/correlations involved with the feature.
- The model is trained on smaller dataset with 19999 training samples.
- APS Data set involves the following pre-processing steps:
    a. Data parsing is essential to import the data into the system (or working environment). Since my implementation is in Python, read_csv() from pandas library is helpful for importing the data in excel to Dataframe type.
    b. All the features are numeric data representation, hence type conversion is not essential.
    c. Missing values in each of 170 features of smaller training data are as follows.

[0, **15489**, 1149, 4975, 865, 865, 208, 208, 208, 208, 208, 208, 208, 208, 208, 208, 239, 231, 231, 1515, 242, 231, 242, 205, 242, 205, 937, 231, 231, 231, 864, 864, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 213, 213, 213, 213, 213, 213, 213, 213, 213, 213, 213, 239, 938, 939, 865, 864, 242, 242, 205, 205, 7682, 9107, 13177, 14643, **15426**, **15912**, **16259**, **16420**, 264, 55, 254, 254, 1095, 160, 937, 1487, 264, 1096, 246, 864, 4975, 4975, 4975, 120, 120, 120, 3207, 3315, 213, 213, 213, 213, 213, 213, 213, 213, 213, 213, 213, 4975, 937, 254, **15489**, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 4640, 4640, 4640, 4640, 4640, 4640, 4640, 4640, 4640, 864, 938, 1365, 1365, 1365, 1364, 1365, 1364, 1365, 1365, 254, 938, 939, 939, 939, 939, 939, 939, 939, 937, 937, 937, 937, 1365, 3437, 3207, 206, 206, 206, 206, 206, 206, 206, 206, 206, 206, 937, 937]

d. **Remove Features:** Few features that contain more than 75% of 19999 samples are removed.

   List of removed columns with missing value greater than 15000 (75% of 19999) are:

Features	Missing values
ab_000	15489
bo_000	15426
bp_000	15912
bq_000	16259
br_000	16420
cr_000	15489

   Total number of missing values after removing 6 columns is **189548** which has been reduced from **284543**

e. **Data filing**: The missing values are compensated using the feature mean using SimpleImputer() function for 164 features. This method is efficient since:
   i. Using k-nearest neighbour method might not be useful in case were 169 features having missing values out of 170 features.
   ii. Filling with respect to only each feature/column is better compared to filing data based/ dependent on other features.
   iii. Filing using class mean is a not a good approach since it is an imbalanced data set.

f. **Removing the sample**: There are few training samples (<10) which have 169 missing feature values. Since it is very small, it can be neglected. Training samples(N) are more important since N >>>Number of learning variables to model the system efficiently and to have a generalized structure. Hence, cannot afford to lose a training sample.

g. Standardizing/Normalizing the feature set is an important task. In order to interpret the results efficiently since different features have different absolute values and units of comparison. The mean and standard deviation are calculated over all class labels on the training dataset using StandardScaler.fit() command. The mean and standard deviation of training dataset is used for standardizing test dataset using the StandardScaler.transform() command of sklearn library.

Different pre-processing techniques was tested for different pattern recognition methods. The results are tabulated in section 4.

3.3. Dataset Methodology

3.3.1 Compensation for unbalanced data

- APS dataset is an unbalanced data as shown in Table 3.3.1.

Table 3.3.1 Tabulation of label wise samples

Label	pos	neg
Number of training samples	333	19666
Percentage of training samples (%)	1.665083	98.334916
Class	Minority Class	Majority class

- Since the difference between the two class samples is high, data has to be balanced before training the model
- Training the model using unbalanced data will yield more favorable conditions for the choice of majority class compared to minority class while testing for unknown data.
- Synthetic Minority Oversampling Technique (SMOTE) is adopted to oversample the minority class sample using oversampling_SMOTE() from imblearn library.
- Procedure for SMOTE:

  a. Identification of a point and its nearest neighbor.

  b. Difference between the two point is multiplied by a random number in the range of [0,1]

  c. Identification of a new point on the line is found by adding the (b) result to the feature vector

  d. For every minority class, approximately 59 sample are generated with the same procedure as mentioned above. Thus, the total number of samples is 39332.
- This method is preferred compared to other method:

  a. Collection of data is not possible. Since it is tedious.

  b. Using a set of classifiers: The model is trained with different non-overlapping majority class and minority class. The classifier output is aggregated to one output based on majority voting. As the difference between the class samples increases, the need of using more classifiers increases, which is not efficient.

Comparison between using unbalanced data and using balanced data (smote technique) is discussed in the results section 3.5.

3.3.3 Dataset Usage

- Total of 19999 training samples were considered for training (unbalanced data). Out of which 333 belongs to “pos” class and 19666 belongs to “neg” class.
- To oversample the minority class label “pos” samples, SMOTE is adopted by generating approximately 59 samples for every minority class training sample.
- Generated Balanced dataset contains 39322 training samples, out of which 19999 training sample belongs to each “pos” and “neg” class.
- Test data contains 16000 samples, out of which 375 sample belongs to “pos” label and 15625 sample belongs to “neg” label.
- Combinations of unbalanced (or balanced) data sets with standardization are used for training the model.
- Testing data with standardization using the corresponding training sample mean and standard deviation are used while testing data.
- 3 classifiers: Random Forest Classifier, Linear Regression, KNN (K nearest neighbor), are trained and tested under various conditions.
- Corresponding training and testing accuracy, confusion matrix, F score are tabulated.
- Final inference is drawn from each of the results.

3.4. Training and Classification

3.4. Training Process

For the machine learning method or model, the following steps are involved:

- Balanced training dataset (using SMOTE) consists of 39332 with 19666 samples in each class.
- Test set consists of 16000 samples with 375 “pos” samples and 15625 “neg” samples.
- Training set is differentiated from test set before training the model using hypothesis set.
- Hypothesis set is created by selecting a range of hyper parameter values of a classifier.
- Balanced training set is further divided into “training set” and “validation set” with ratio 70:30.
- Validation set is used in cross validation for choosing the best estimator (hypothesis) amongst different hyper parameters of a classifier (hypothesis set) and to avoid overfitting/underfitting.
- The model is trained on training set and tested on validation set, to choose the best hypothesis set based on the total cost incurred when tested on validation set. (We are interested in minimizing the total cost prediction of a model – goal of the project).
- Once the best hypothesis set is chosen, error on testing set is estimated before which test set is standardized by mean and standard deviation of balanced training dataset.
- Accuracy, F1 score, confusion matrix, ROC is estimated for balanced training and test set.
- The main objective is to minimize False positive and false negative samples/instances to minimize the total cost. Since the cost for false negative is high, recall must be higher in order to minimize the cost.
- Thus, ROC/ precision_recall curve can be used to choose a threshold such that the recall is high and the total cost is minimized. Validation set is used for selecting threshold using cross validation.

3.5. Model Selection and Comparison of Results

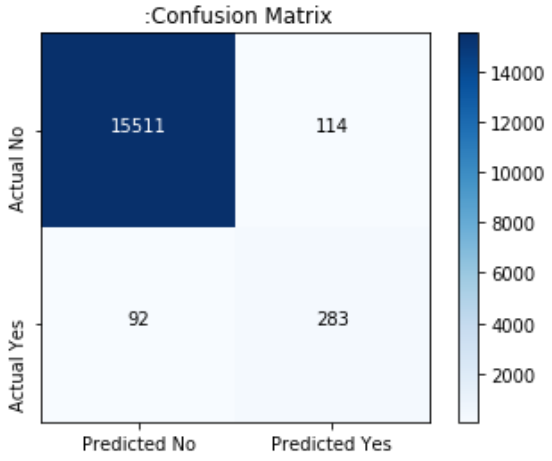
1. Random Forest Classifier

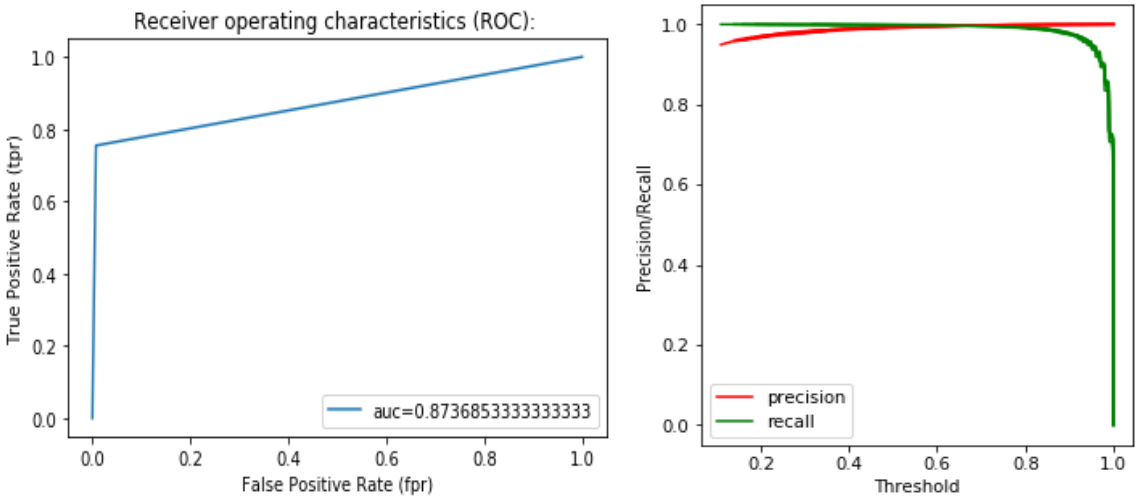
   (Reference/Source: https://scikit-learn.org/stable/modules/ensemble.html#forest)

   A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).
   In random forests each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.
   When splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max_features based on the impurity measure.
   The purpose of these two sources of randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.
   Random Forest was chosen since there are enough data samples to learn the model.

   Case 1:

   Data filing for missing values: **Mean** of the feature
   Standardization: Normalization (mean=0, standard deviation=1)
   Hyper Parameters/ Hypothesis Set : Max depth= [10,20,30]
   n_estimators= [50,100,200,250,300]
   Best Hyperparameter/hypothesis set: max_depth=30, n_estimator=100
   Testing F1 Score: 0.987
   Type 1 error (False Positive) = 114
   Type 2 error (False Negative) = 92
   Total cost = 47140





Confusion matrix after adjusting threshold > 0.1 by observing Precision-Recall Curve

[[14669 956]

[9 366]]

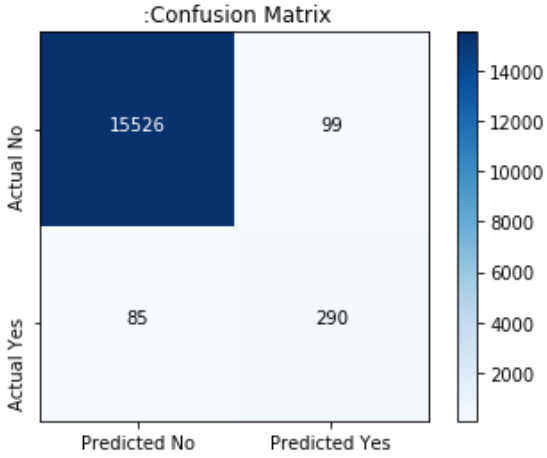
Type 1 error (False Positive) = 956

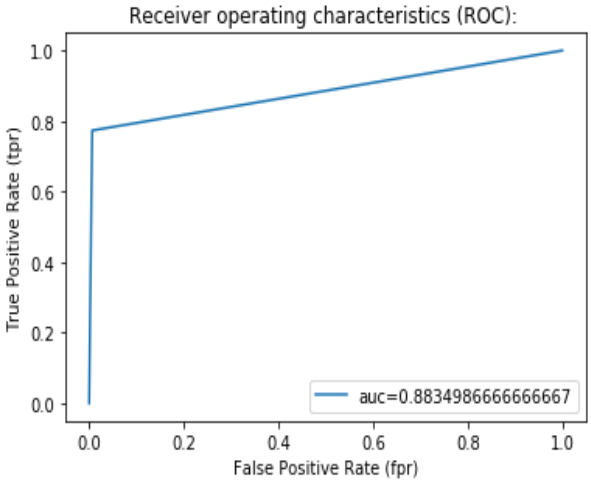
Type 2 error (False Negative) = 9

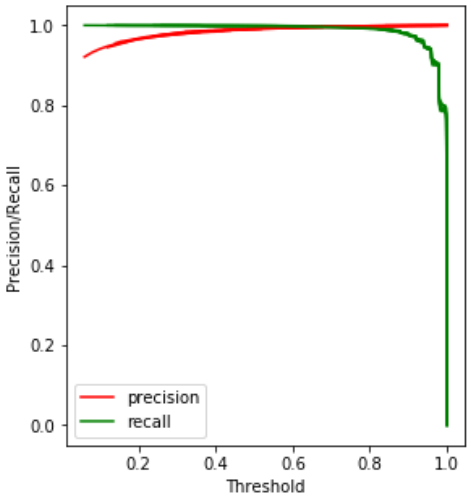
Total cost = **14060**

Case 2:

Data filing for missing values: **Median** of the feature
Standardization: Normalization (mean=0, standard deviation=1)
Hyper Parameters/ Hypothesis Set : Max depth= [10,20,30]
n_estimators= [50,100,200,250,300]
Best Hyperparameter/hypothesis set: max_depth=30, n_estimator=50
Testing F1 Score: 0.989
Type 1 error (False Positive) = 99
Type 2 error (False Negative) = 85
Total cost = 43490







Confusion matrix after adjusting threshold > 0.2 by observing Precision-Recall Curve

[[15167 458]

[21 354]]

Type 1 error (False Positive) = 458

Type 2 error (False Negative) = 21

Total cost = 15080

2. Logistic Regression

(Reference/Source: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

Logistic regression implementation can fit binary, One-vs-Rest, or multinomial logistic regression with optional, or Elastic-Net regularization.

As an optimization problem, binary class penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i(X_i^T w + c)\right) + 1\right)$$

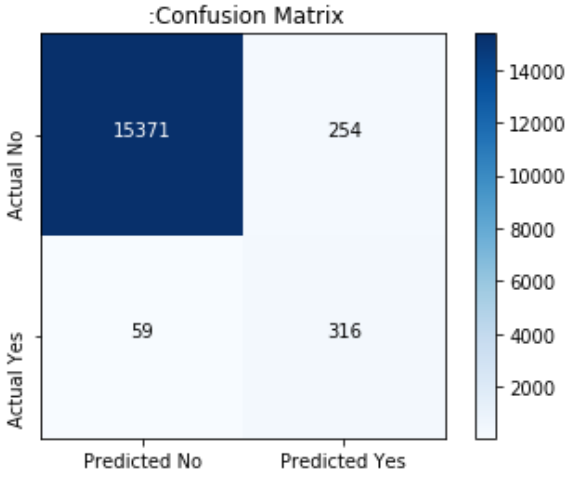
Similarly, regularized logistic regression solves the following optimization problem:

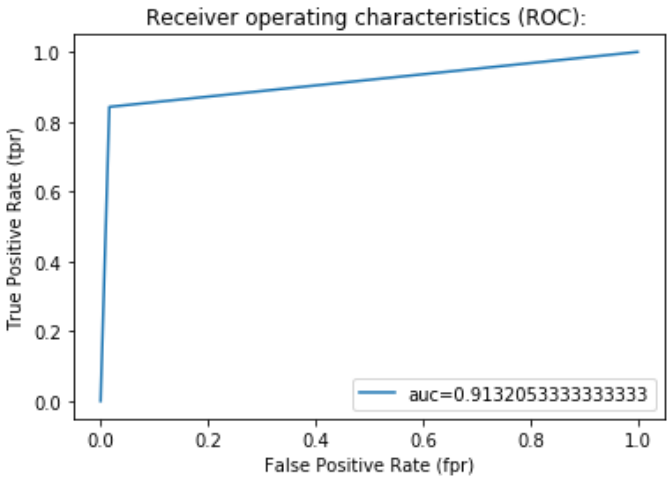
$$\min_{w,c} |w|_1 + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i(X_i^T w + c)\right) + 1\right)$$

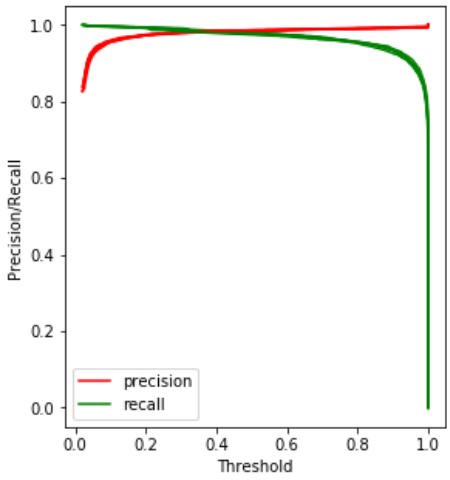
Logistic Regression was selected because it is a simple implementation in classification problem.

Case 1:

Data filing for missing values: **Mean** of the feature
Standardization: Normalization (mean=0, standard deviation=1)
Hyper Parameters/ Hypothesis Set :
C= [10**-3,10**-2,10**-1,1,10**1,10**2,10**3]
penalty=['l1','l2']
Best Hyperparameter/hypothesis set:
C= 1000
penalty=['l1']
Testing F1 Score: 0.982
Type 1 error (False Positive) = 254
Type 2 error (False Negative) = 59
Total cost = 32040





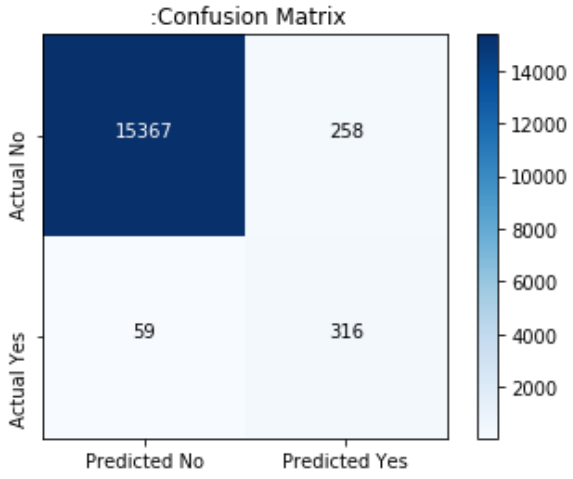


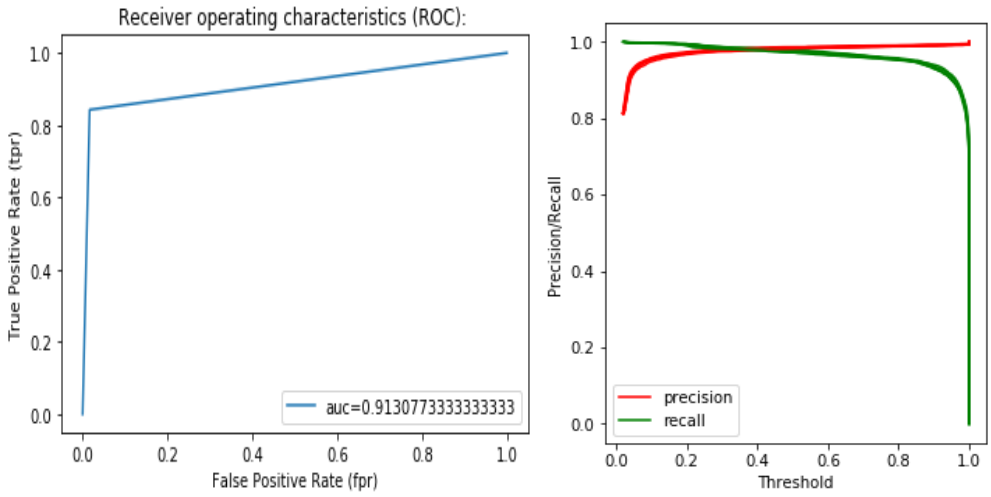
Confusion matrix after adjusting threshold > 0.1 by observing Precision-Recall Curve

[[14907 718]
[38 337]]
Type 1 error (False Positive) = 718
Type 2 error (False Negative) = 38
Total cost = **26180**

Case 2:

Data filing for missing values: **Median** of the feature
Standardization: Normalization (mean=0, standard deviation=1)
Hyper Parameters/ Hypothesis Set :
C= [10**-3,10**-2,10**-1,1,10**1,10**2,10**3]
penalty=['l1','l2']
Best Hyperparameter/hypothesis set:
C= 10
penalty=['l1']
Testing F1 Score: 0.982
Type 1 error (False Positive) = 258
Type 2 error (False Negative) = 59
Total cost = 32080





Confusion matrix after adjusting threshold > 0.1 by observing Precision-Recall Curve

[[14855 770]

[38 337]]

Type 1 error (False Positive) = 770

Type 2 error (False Negative) = 38

Total cost = **26700**

3. K Nearest Neighbour classifier

Neighbors-based classification is a type of *instance-based learning* or *non-generalizing learning*. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

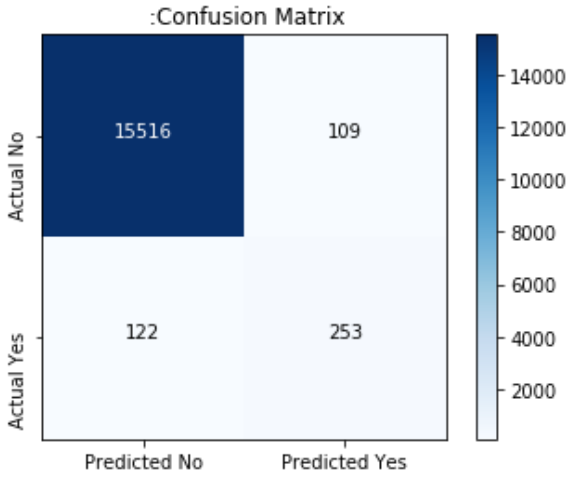
KNeighborsClassifier implements learning based on the nearest neighbors of each query point, where is an integer value specified by the user.

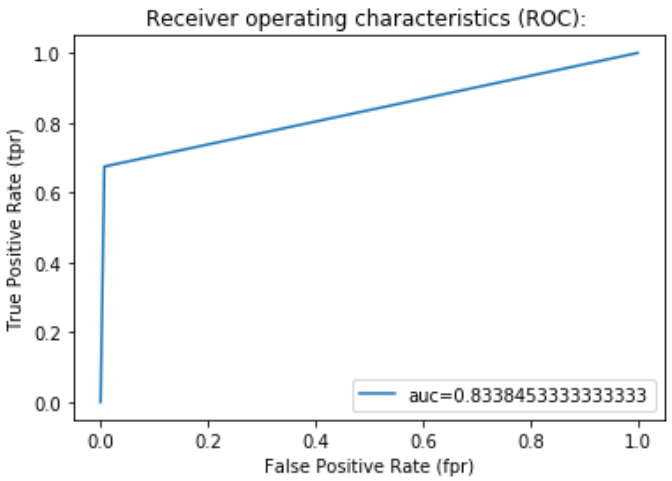
The k-neighbors classification in **KNeighborsClassifier** is the most commonly used technique. The optimal choice of the value is highly data-dependent: in general a larger suppresses the effects of noise, but makes the classification boundaries less distinct. For high-dimensional parameter spaces, this method becomes less effective due to the so-called “curse of dimensionality”.

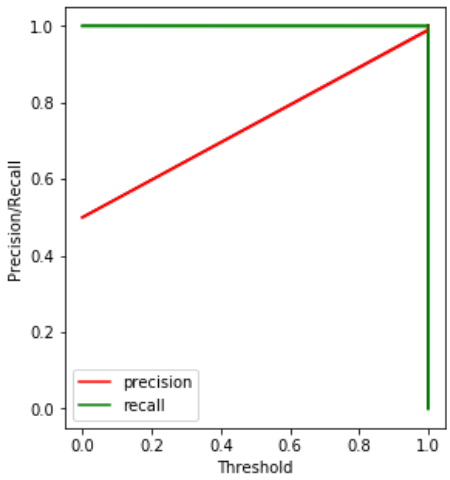
The basic nearest neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors.

Case 1:

Data filing for missing values: **Mean** of the feature
Standardization: Normalization (mean=0, standard deviation=1)
Hyper Parameters/ Hypothesis Set :
Number of neighbors: [1,5,10,50,100,500,1000,10000]
Best Hyperparameter/hypothesis set:
Number of neighbors: 1
Testing F1 Score: 0.985
Type 1 error (False Positive) = 109
Type 2 error (False Negative) = 122
Total cost = 62090





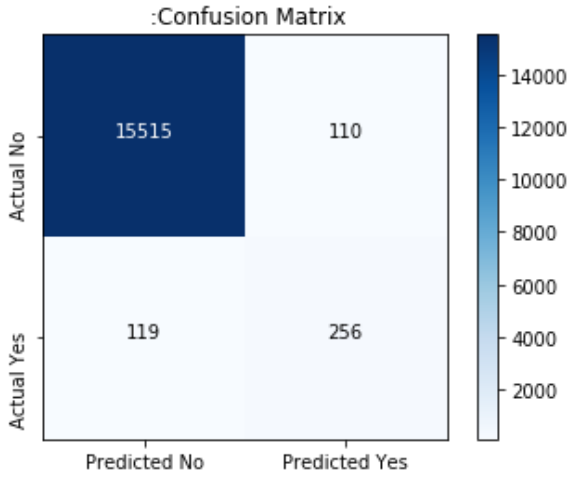


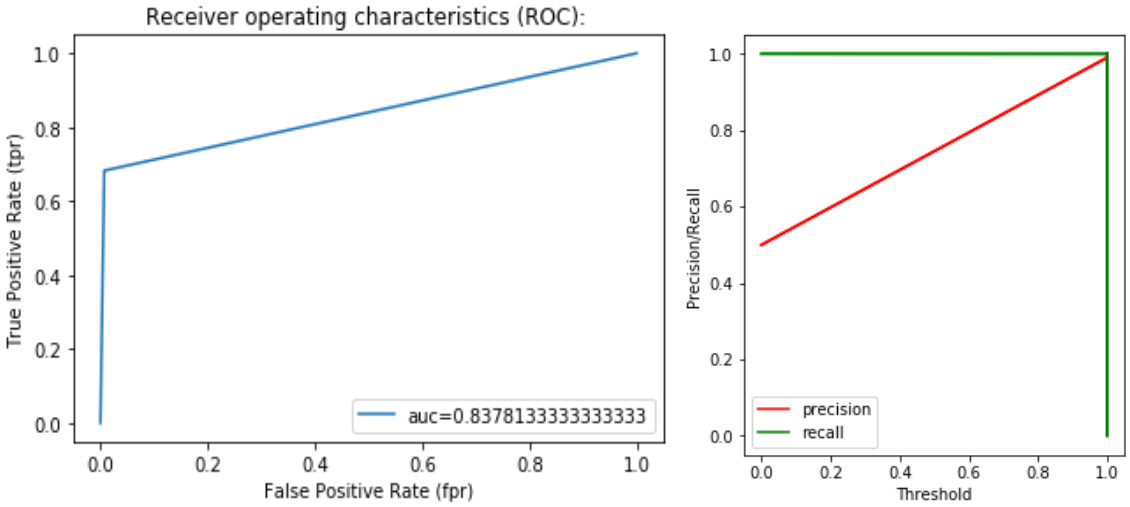
Confusion matrix after adjusting threshold > 0.1 by observing Precision-Recall Curve

[[15516 109]
[122 253]]
Type 1 error (False Positive) = 109
Type 2 error (False Negative) = 122
Total cost = 62090

Case 2:

Data filing for missing values: **Median** of the feature
Standardization: Normalization (mean=0, standard deviation=1)
Hyper Parameters/ Hypothesis Set :
Number of neighbors: [1,5,10,50,100,500,1000,10000]
Best Hyperparameter/hypothesis set:
Number of neighbors: 1
Testing F1 Score: 0.986
Type 1 error (False Positive) = 110
Type 2 error (False Negative) = 119
Total cost = 60600





Confusion matrix after adjusting threshold > 0.1 by observing Precision-Recall Curve

[[15515 110]

[119 256]]

Type 1 error (False Positive) = 110

Type 2 error (False Negative) = 119

Total cost = 60600

4. Final Results and Interpretation

Classifier	Missing Value Imputer method	Test F1 score	Total cost after model selection	Total cost after changing threshold by observing precision_recall curve
Random Forest Max_depth = 20, N_estimator = 500	Mean	0.987	47140	14060
Random Forest Max_depth = 20, N_estimator = 500	Median	0.989	43490	15080
Logistic Regression C = 1000 Penalty = 'l1'	Mean	0.982	32040	26180
Logistic Regression C =10 Penalty = 'l1'	Median	0.982	32080	26700
KNN k=1	Mean	0.985	62090	62090
KNN k= 1	Median	0.986	60600	60600

Interpretation:

- By changing the threshold of the predicted probability, the total cost of the prediction model has drastically decreased from just the best parameter selection to adjusting the threshold in the precision recall curve in case of Random Forest classifier and Logistic Regression
- The Imputer method() used for filling the missing values can be either filled using mean of the feature or median of the feature. From the above table, it can be observed that the data filled with mean of feature performs better compared to data filled with median of feature for the Random Forest classifier and logistic regression.
- In case of K nearest neighbor, the pattern observed is opposite for total cost with respect to the values of mean, median imputer and total cost after and before adjusting the threshold of ROC. I think the reason for low performance is that, since 169 features out of 170 features are filled with mean/median, the uncertainity of most of the features for a given data sample has caused the performance to be low. (Error has propogated

throughout the sample/instances, and distance between those error samples, may not be a good metric to classify the samples).

- Random forest, which treats each feature differently, according to the feature distribution in order to build a decision tree based on the impurity gain, thus performs better. Since the missing values are filled based on its own feature.
- The intension of keeping recall high compared to precision has worked out well to minimize the total cost of production, since the cost of false negative was high and the difference between the cost after adjusting the threshold is evident for Random forest classifier and logistic regression.

5. Contributions of each team member

Individual project

**Note: For each model selected, a range of hyper parameters were selected to be a hypothesis set. Using validation set, the best hypothesis was selected (code has been written on own). Based on the best hyper parameter obtained, confusion matrix, F1 score, ROC, total cost prediction of a model are estimated. Precision Recall curves are plotted using own code. By adjusting the threshold of the predicted probability of a model, the total cost prediction of model, F1 score, confusion matrix are obtained and evaluated by using own code.

Each model has taken more than 2 hours to model completely on Google colaboratory. Hence, comparision could not be performed on more classifiers.

6. Summary and conclusions

The course project has helped me in gaining a hands-on experience working with real datasets which are far ahead from the theoretical assumptions. Interpretation/analysis of pre-processing, feature selection (extraction), training model, classification, model hyper-parameter tuning was helpful.

7. References

[1] https://scikit-learn.org/stable/modules.com

8. Apendix

Main file consists for preprocessing using mean imputer(), classifier, training, hyper parameter tuning etc

Main_continued file consists for preprocessing using median imputer(),, classifier, training, hyper parameter tuning etc