DRIVER INSURANCE CLAIM PREDICTION

Yash Hatekar (yhatekar)

Rishika Samala (rsamala)

Diksha Singh (singhdi)

MOTIVATION

- Auto Insurances have become a key part of owning a vehicle nowadays. It has become a necessity for vehicle owners.
- But despite being a necessity people don’t weigh the importance of getting the best insurance suitable for them. Generally people try to find the cheapest possible insurance to cover their vehicle.
- The traditional way of insurance brokers handling the insurance has reduced and online purchases of insurances has increased. Therefore the ability of companies to contact the customers directly has reduced and the best insurance should be quoted at a good reasonable price for a particular customer based on his safety and priorities.
- With this project we want to change this paradigm by giving companies an idea about how likely a person is to file a claim, so that the companies can come up with more lucrative and safer insurance for their customers which is beneficial to both of them.

DATA FROM KAGGLE

- Number of Instances: 595212
- Number of Features: 59
- **Features Description:**
    - bin - features with binary values (0 or 1),
    - cat - features with categorical values
    - Features with tags as ‘ind’, ‘reg’, ‘car’, ‘calc’:
        - ind - personal info of customer (name, etc.)
        - reg - region/location info customer.
        - calc - calculated features.

PRIOR WORK:

- An analysis of customer retention and insurance claim patterns using data mining: A case study. [1]
- Use of optimized Fuzzy C-Means clustering and supervised classifiers for automobile insurance fraud detection.[2]
- Research on Probability-based Learning Application on Car Insurance Data.[3]
- A proposed model to predict auto insurance claims using machine learning techniques.[4]

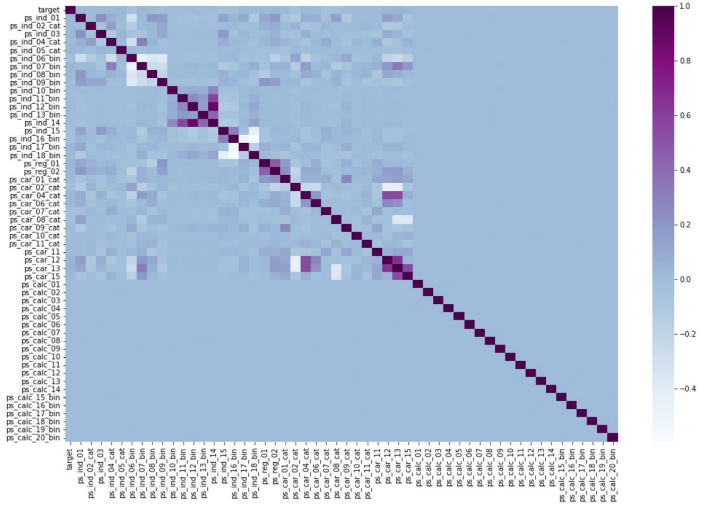
[1] https://www.tandfonline.com/doi/abs/10.1057/palgrave.jors.2600941
[2] https://www.sciencedirect.com/science/article/pii/S1319157817301672?via%3Dihub
[3] https://www.researchgate.net/publication/322478575_Research_on_Probability-based_Learning_Application_on_Car_Insurance_Data
[4] http://www.jatit.org/volumes/Vol98No22/8Vol98No22.pdf

PROPOSED APPROACH:

1. **Data Preprocessing and Exploratory Data Analysis:**
   a. Handling NaN values
   b. Handling highly correlated features
   c. Encoding Categorical Features
   d. Splitting data into train and test datasets
   e. Feature Scaling
   f. Handling Imbalanced Data
2. **Training using different algorithms:** Applying different algorithms of Logistic regression, Support Vector Classification, XGBoost, Adaboost, K Nearest Neighbors, Gaussian Naive Bayes, Bagging Classification techniques.
3. **Evaluation Metrics:** Defined and analyzed accuracy, precision, recall, F1 score, ROC characteristics of test data on trained model.
4. **Hyper parameters Tuning:** Tuned hyper parameters by experimenting with different values to reduce the error and improve the prediction.
5. **Results:** Compared the results and figured which model performs the best.

DATA PREPROCESSING

- **Handling NaN values:** Removed features having very high number of NaN values. Removed instances having NaN for other attributes.
- **Handling correlated features:** There are no features which are very highly correlated so that they can be combined or one of them removed. Also, features with zero correlation are removed as they dont give any information regarding data. Dropped all binary calc features from data after finding correlation.



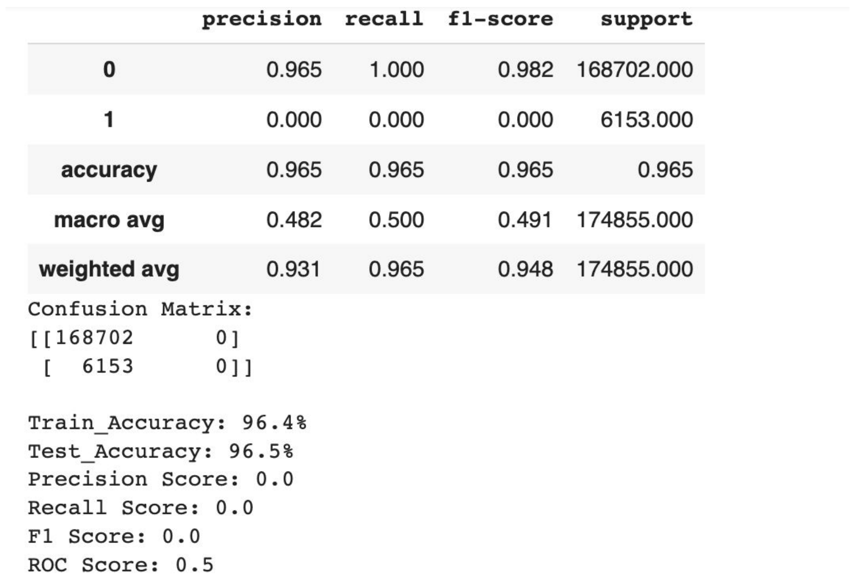
DATA PREPROCESSING

- **Encoding Categorical Features:** Applied one hot encoding to categorical attributes before using them for training. After one hot encoding the total number of attributes changed from 34 to 181.
- **Splitting data into train and test datasets:** Used Test Train split to divide the data into training and testing datasets with 70% and 30% of data division.
- **Feature Scaling:** Performed Standardization of feature data by making data with zero mean and unit variance. This helps in the model perceiving all the attributes to be same and not prioritizing any one based on different data ranges.

TRAINING DATA USING DIFFERENT CLASSIFICATION MODELS:

LOGISTIC REGRESSION ON UNSAMPLED DATA:

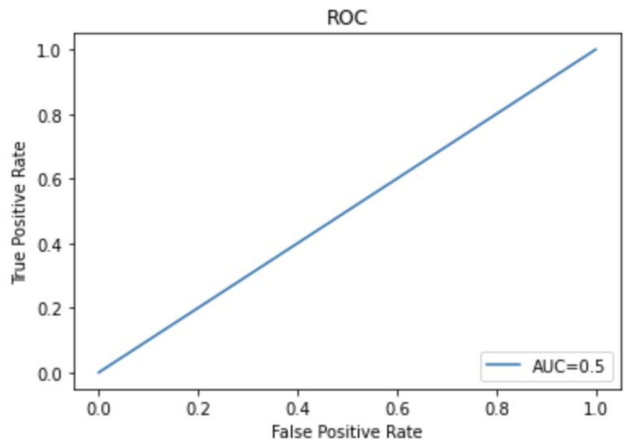
- Applying Logistic Regression with penalty l2, lbfgs solver and with maximum iterations as 100 we observed the metrics as follows:



	precision	recall	f1-score	support
0	0.965	1.000	0.982	168702.000
1	0.000	0.000	0.000	6153.000
accuracy	0.965	0.965	0.965	0.965
macro avg	0.482	0.500	0.491	174855.000
weighted avg	0.931	0.965	0.948	174855.000

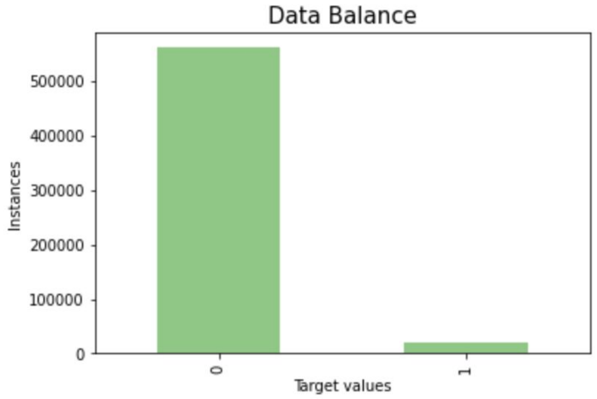
```
Confusion Matrix:
[[168702      0]
 [  6153      0]]

Train_Accuracy: 96.4%
Test_Accuracy: 96.5%
Precision Score: 0.0
Recall Score: 0.0
F1 Score: 0.0
ROC Score: 0.5
```

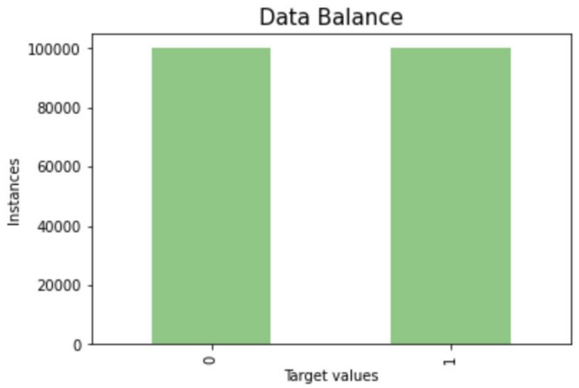


HANDLING IMBALANCED DATA:

- We observed the data is highly imbalanced with target data as 0 as **562105** values and value 1 as **20742**. So we performed upsampling of minority target data and downsampling of majority target data to get equal instances of both classes with 100k instances of each class.



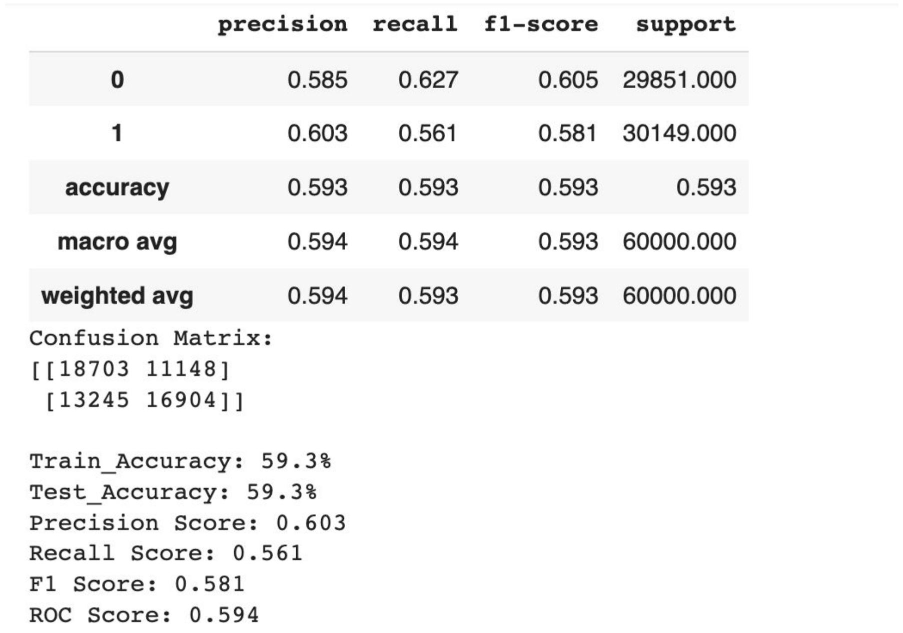
Before Sampling



After Sampling

LOGISTIC REGRESSION ON SAMPLED DATA:

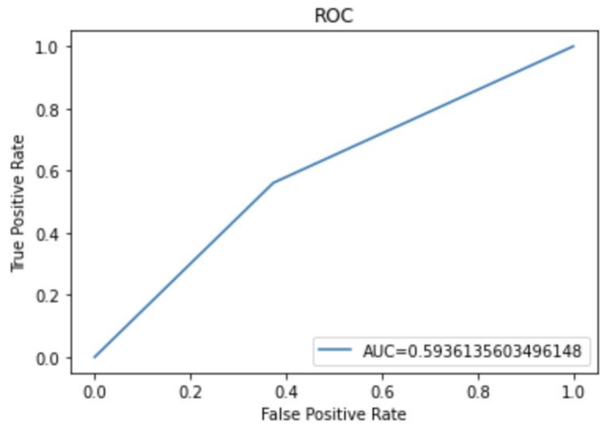
- Applying Logistic Regression with penalty l2, lbfgs solver and with maximum iterations as 100 we observed the metrics as follows:



	precision	recall	f1-score	support
0	0.585	0.627	0.605	29851.000
1	0.603	0.561	0.581	30149.000
accuracy	0.593	0.593	0.593	0.593
macro avg	0.594	0.594	0.593	60000.000
weighted avg	0.594	0.593	0.593	60000.000

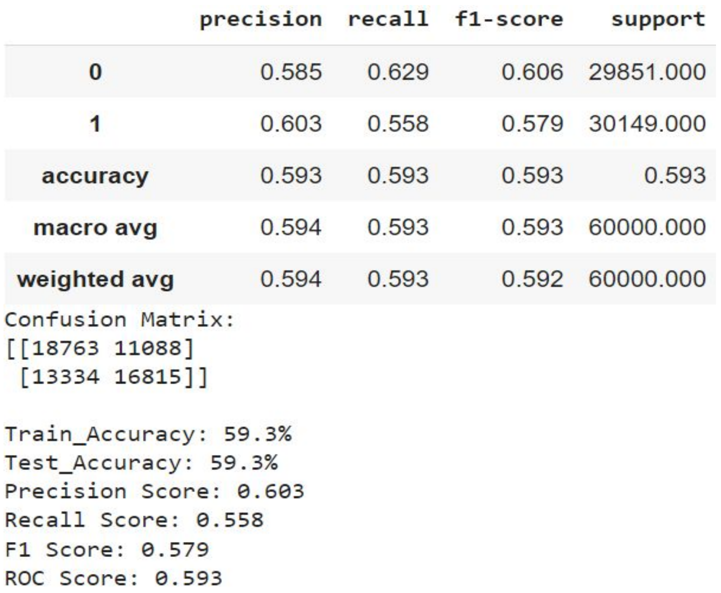
```
Confusion Matrix:
[[18703 11148]
 [13245 16904]]

Train_Accuracy: 59.3%
Test_Accuracy: 59.3%
Precision Score: 0.603
Recall Score: 0.561
F1 Score: 0.581
ROC Score: 0.594
```



LINEAR SUPPORT VECTOR CLASSIFICATION:

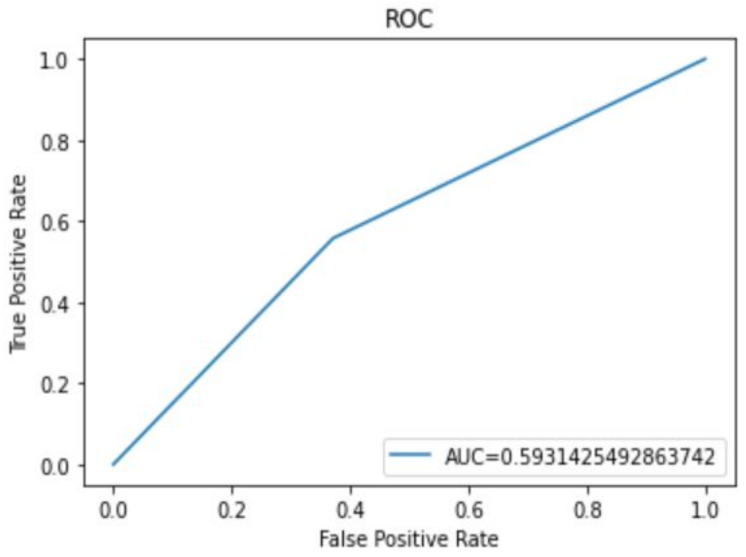
- We used Grid Search CV for choosing best hyperparameters for Linear SVC and found at max iterations of 2 it was giving good F1 score. With this trained model we found the results as follows:



	precision	recall	f1-score	support
0	0.585	0.629	0.606	29851.000
1	0.603	0.558	0.579	30149.000
accuracy	0.593	0.593	0.593	0.593
macro avg	0.594	0.593	0.593	60000.000
weighted avg	0.594	0.593	0.592	60000.000

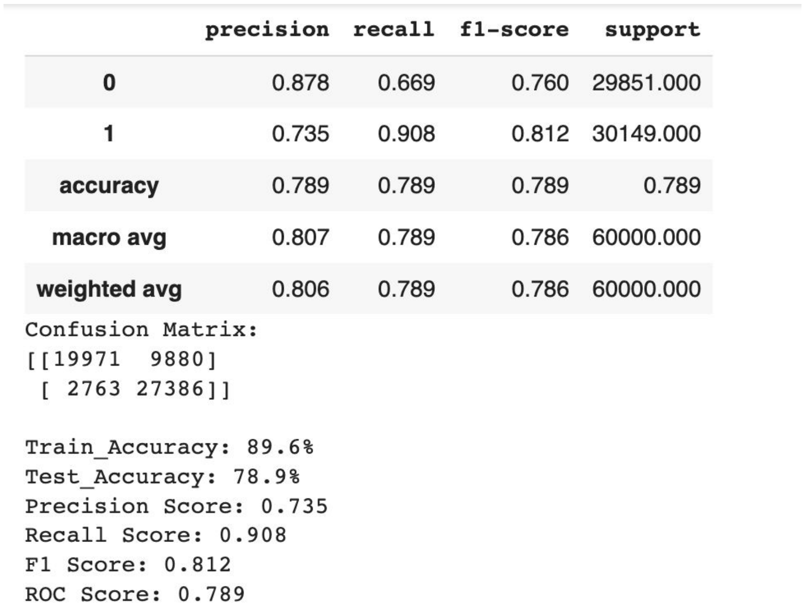
```
Confusion Matrix:
[[18763 11088]
 [13334 16815]]

Train_Accuracy: 59.3%
Test_Accuracy: 59.3%
Precision Score: 0.603
Recall Score: 0.558
F1 Score: 0.579
ROC Score: 0.593
```



K-NEAREST NEIGHBOUR

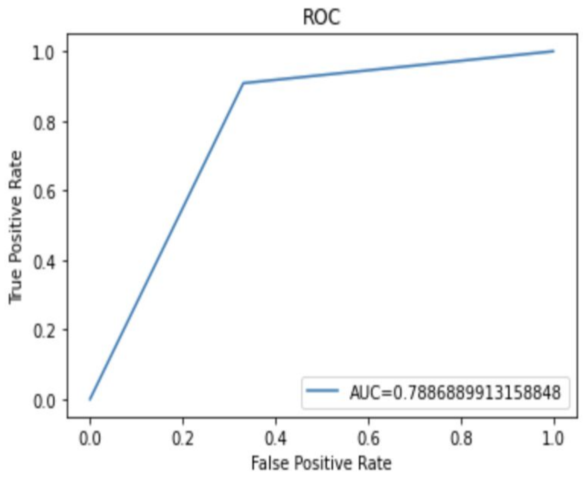
- We applied K nearest neighbors model with number of neighbors as 3, distance metric as Minkowski, weights as uniform, leaf size as 30. The metrics are as follows:



	precision	recall	f1-score	support
0	0.878	0.669	0.760	29851.000
1	0.735	0.908	0.812	30149.000
accuracy	0.789	0.789	0.789	0.789
macro avg	0.807	0.789	0.786	60000.000
weighted avg	0.806	0.789	0.786	60000.000

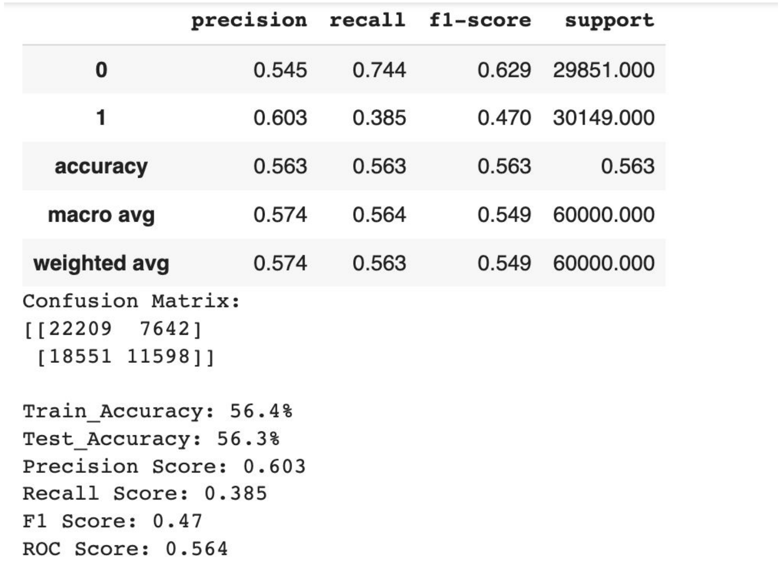
```
Confusion Matrix:
[[19971  9880]
 [ 2763 27386]]

Train_Accuracy: 89.6%
Test_Accuracy: 78.9%
Precision Score: 0.735
Recall Score: 0.908
F1 Score: 0.812
ROC Score: 0.789
```



GAUSSIAN NAIVE BAYES CLASSIFIER

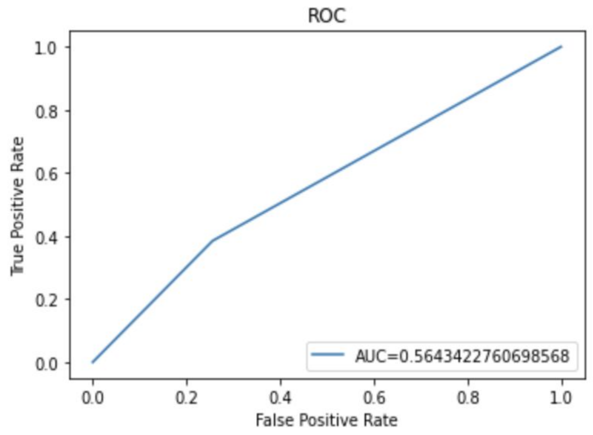
- Applied Naive Bayes Classifier with variance smoothing as 1e-09 and observed metrics as follows:



	precision	recall	f1-score	support
0	0.545	0.744	0.629	29851.000
1	0.603	0.385	0.470	30149.000
accuracy	0.563	0.563	0.563	0.563
macro avg	0.574	0.564	0.549	60000.000
weighted avg	0.574	0.563	0.549	60000.000

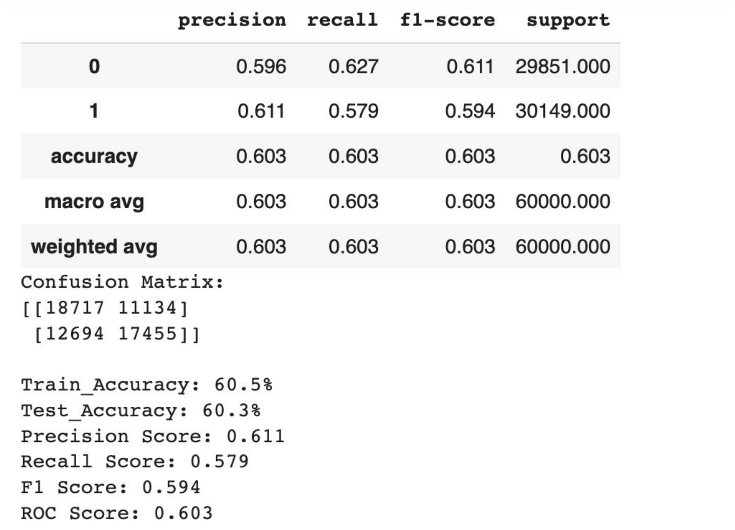
```
Confusion Matrix:
[[22209  7642]
 [18551 11598]]

Train_Accuracy: 56.4%
Test_Accuracy: 56.3%
Precision Score: 0.603
Recall Score: 0.385
F1 Score: 0.47
ROC Score: 0.564
```



XGBOOST CLASSIFIER

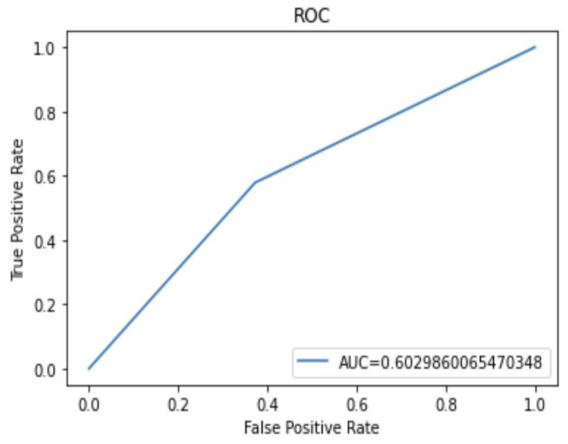
- We performed XGBoost Classification and observed the results as follows:



	precision	recall	f1-score	support
0	0.596	0.627	0.611	29851.000
1	0.611	0.579	0.594	30149.000
accuracy	0.603	0.603	0.603	0.603
macro avg	0.603	0.603	0.603	60000.000
weighted avg	0.603	0.603	0.603	60000.000

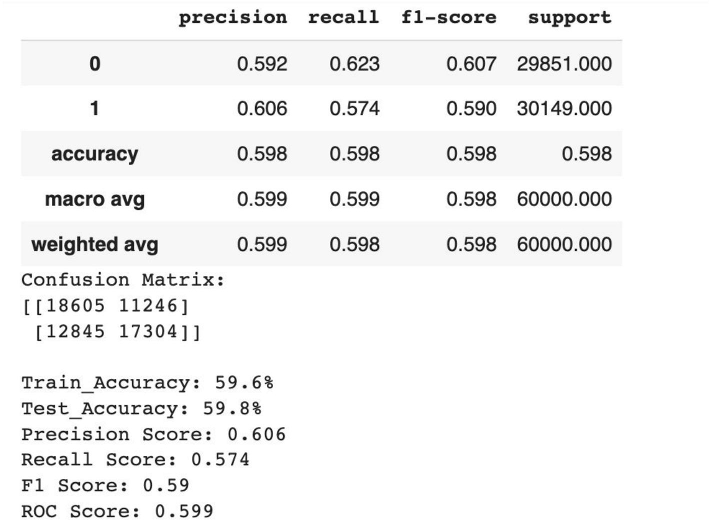
```
Confusion Matrix:
[[18717 11134]
 [12694 17455]]

Train_Accuracy: 60.5%
Test_Accuracy: 60.3%
Precision Score: 0.611
Recall Score: 0.579
F1 Score: 0.594
ROC Score: 0.603
```



ADABOOST CLASSIFICATION(DECISION TREE)

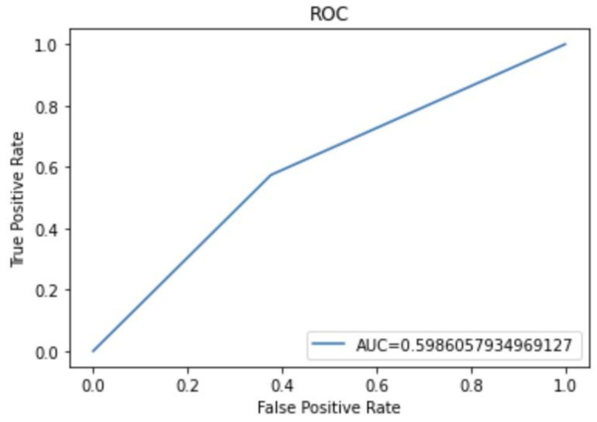
- Performing Adaboost classification using Decision tree as base estimator with estimators as 100 and learning rate as 1.0. The metrics are observed as follows:



	precision	recall	f1-score	support
0	0.592	0.623	0.607	29851.000
1	0.606	0.574	0.590	30149.000
accuracy	0.598	0.598	0.598	0.598
macro avg	0.599	0.599	0.598	60000.000
weighted avg	0.599	0.598	0.598	60000.000

```
Confusion Matrix:
[[18605 11246]
 [12845 17304]]

Train_Accuracy: 59.6%
Test_Accuracy: 59.8%
Precision Score: 0.606
Recall Score: 0.574
F1 Score: 0.59
ROC Score: 0.599
```



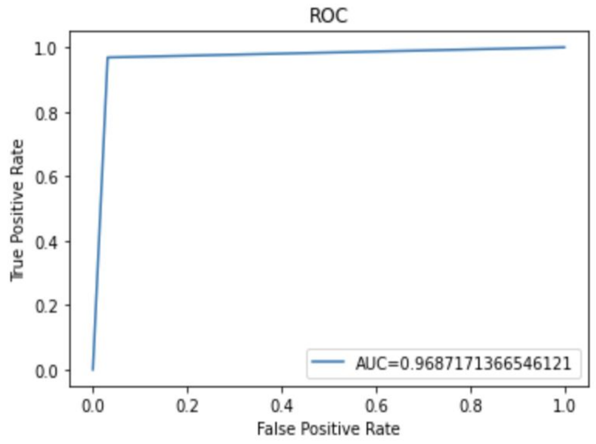
ADABOOST (RANDOM FOREST)

- Performing Adaboost classification using Random Forest as base estimator with estimators as 100 and learning rate as 1.0. The metrics are observed as follows:

	precision	recall	f1-score	support
0	0.968	0.969	0.969	29851.000
1	0.969	0.969	0.969	30149.000
accuracy	0.969	0.969	0.969	0.969
macro avg	0.969	0.969	0.969	60000.000
weighted avg	0.969	0.969	0.969	60000.000

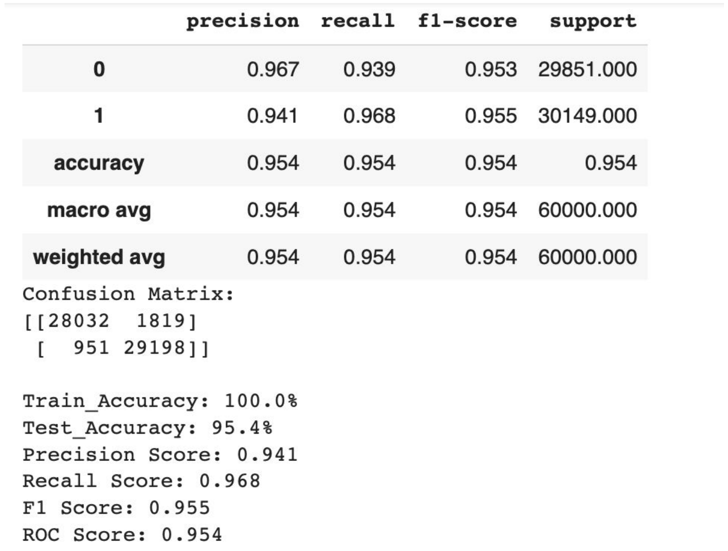
```
Confusion Matrix:
[[28920   931]
 [  946 29203]]

Train_Accuracy: 100.0%
Test_Accuracy: 96.9%
Precision Score: 0.969
Recall Score: 0.969
F1 Score: 0.969
ROC Score: 0.969
```



BAGGING CLASSIFIER (DECISION TREE)

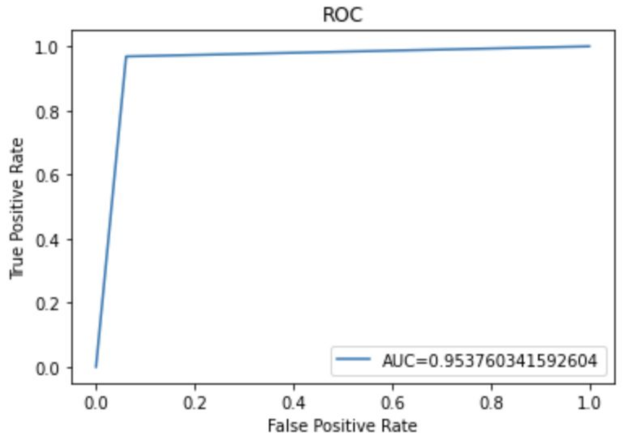
- We tried ensemble bagging classifier with base estimator as Decision Tree and number of estimators as 100, max samples as 1.0. We observed the metrics as follows:



	precision	recall	f1-score	support
0	0.967	0.939	0.953	29851.000
1	0.941	0.968	0.955	30149.000
accuracy	0.954	0.954	0.954	0.954
macro avg	0.954	0.954	0.954	60000.000
weighted avg	0.954	0.954	0.954	60000.000

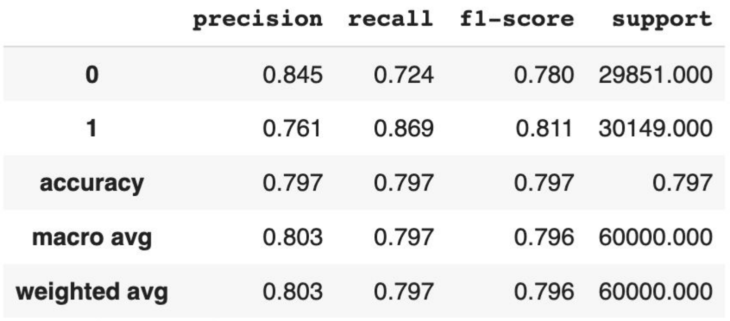
```
Confusion Matrix:
[[28032  1819]
 [  951 29198]]

Train_Accuracy: 100.0%
Test_Accuracy: 95.4%
Precision Score: 0.941
Recall Score: 0.968
F1 Score: 0.955
ROC Score: 0.954
```



ARTIFICIAL NEURAL NETWORK

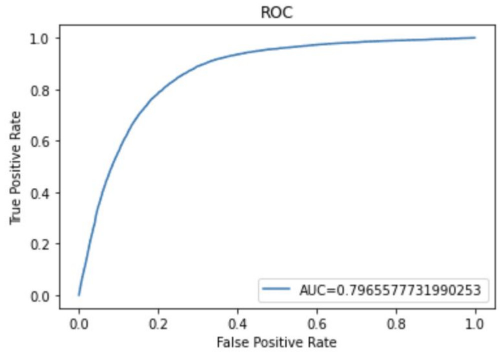
- We experimented with different layers and settled with 4 dense layers and one output layer with 512, 256,128, 64 and 1 number of neurons in each layer respectively. We used initializer ‘RandomNormal’ and used activation ‘Relu’ and ‘Sigmoid’ functions. We observed the metrics and learning curves as follows.

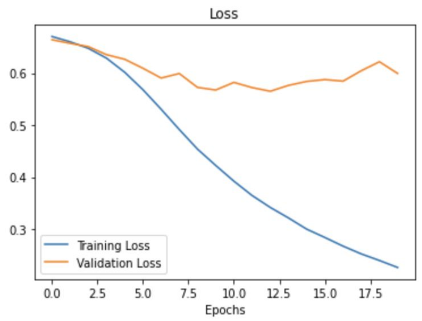


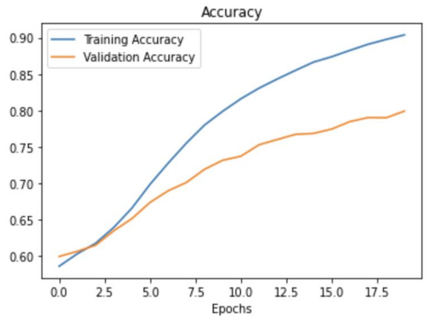
	precision	recall	f1-score	support
0	0.845	0.724	0.780	29851.000
1	0.761	0.869	0.811	30149.000
accuracy	0.797	0.797	0.797	0.797
macro avg	0.803	0.797	0.796	60000.000
weighted avg	0.803	0.797	0.796	60000.000

```
Confusion Matrix:
[[21621  8230]
 [ 3955 26194]]

Train_Accuracy: 90.0%
Test_Accuracy: 79.7%
Precision Score: 0.761
Recall Score: 0.869
F1 Score: 0.811
ROC Score: 0.797
```







Learning Curves

COMPARISON OF RESULTS

Model	**Training Accuracy**	**Testing Accuracy**	**Precision**	**Recall**	**F1 score**	**ROC score**
Logistic Regression	59.3%	59.3%	0.603	0.561	0.581	0.594
Linear SVC	59.3%	59.3%	0.603	0.558	0.579	0.593
XGBoost	60.5%	60.3%	0.611	0.579	0.594	0.603
AdaBoost (Decision Tree)	59.6%	59.8%	0.606	0.574	0.59	0.599
AdaBoost (Random Forest)	100%	96.9%	0.969	0.969	0.969	0.969
KNN	89.6%	78.9%	0.735	0.908	0.812	0.789
Naive Bayes	56.4%	56.3%	0.603	0.385	0.47	0.564
Bagging (Decision Tree)	100%	95.4%	0.941	0.968	0.955	0.954
ANN	90%	79.69%	0.76	0.868	0.811	0.796

CONCLUSION

- In this study, we have shown how ML models can be used to provide an insight for the insurance companies about a customer regarding filing of claims.
- We have presented several techniques which can efficiently analyze insurance claims prediction and compared their performances.
- Using these ML models to predict claim requests, we are providing an insight for a customer which companies can utilize to tailor the auto insurance accordingly.
- This approach can also help the companies to provide faster and more customer tailored insurance.
- With different metrics provided by the models we can get a more transparent view of the decision making algorithm which can be studied to prevent false predictions or any harmful effect for the business.
- Comparisons of the model results showed that **AdaBoost with Random Forest** as base estimator and **Bagging Classifier with Decision Tree** base estimator are the most reliable among all the nine models.

THANK YOU