FINAL CAPSTONE PRESENTATION

Presented By :

Anoosh Kumar
Nidhi Jaiswal
Pranjal Jalota
Rushika Bokde
Sylesh JL

Business Problem-An insurance policy is an agreement between an insurance provider company and policyholder, wherein the company is liable in providing the guaranteed compensation for a specific loss or damage when a certain amount of premium is paid by the policyholder for taking the insurance with that company.
So it’s a challenge for insurance company to estimate how probably the customer will claim insurance for his car based on which they can define insurance policies which are profitable and preferred by customers.

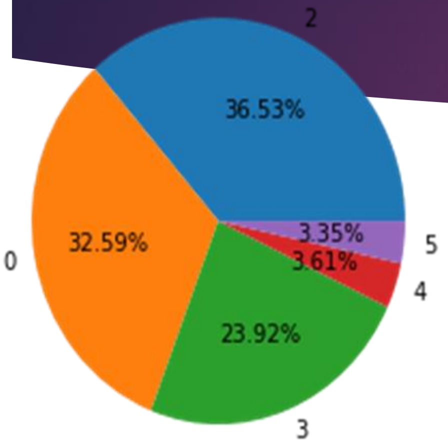
Business Objective :
Our objective is to help the Insurance company to understand the behavior of customers and predict a future trait if a car policy holder will claim his insurance or not based on the insurance data provided.. With the above objective we aim to develop a prediction model that helps the insurance company to understand the car policy holders behavior and classify if the policy holders are likely to claim insurance.

Importance of Problem and Technology used-
Due to the rapid modernization, a surge in the usage of number of cars is seen in the last two decades compelling the insurance companies in utilizing most advanced data techniques in order to predict whether a policyholder opts for the claim or not. In this quest companies started using the machine learning algorithms in their business, Major categories in where the companies currently using ML models include behavior of driver monitoring system and in depth market analysis on Data of insurance claims based on consumer previous behavior.

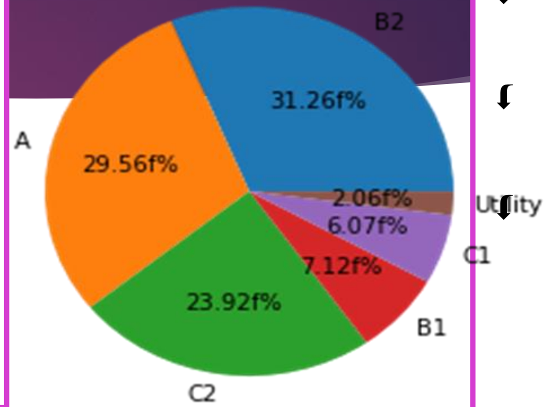
Our Value addition-

- We are using a ML binary classification model to predict the insurance claims.
- We are trying to build a model that can predict the possibility of insurance being claimed based on the description of components, manufacture and model of the car.
- We will design a model that will help the companies to improve their insurance policies based on our claim prediction with providing maximum benefits to insurance companies as well as the policy holder.

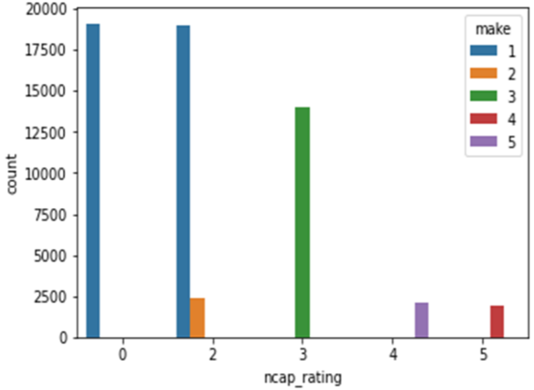
EDA Inferences:



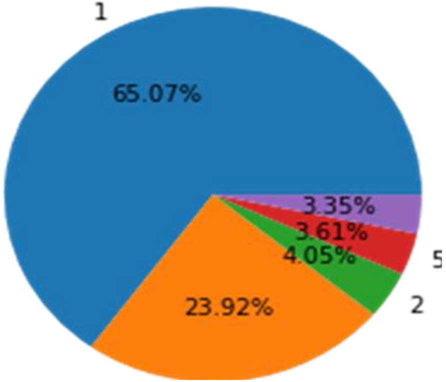
- Most of the cars have an NCAP rating of 2, 0, and 3.
- most cars are not having good safety ratings



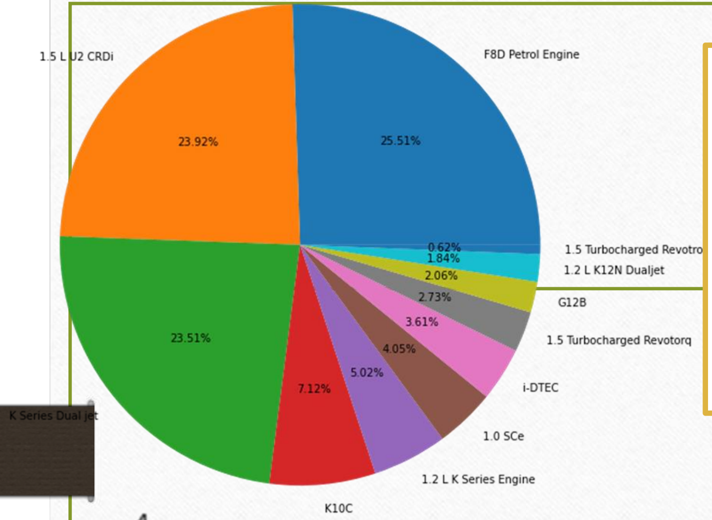
- We can see that most cars are from A, B2 and C2 segment.
- Few are from B1,C1 and utility.
- Hence we can say most of the population goes for a specific segment of cars.



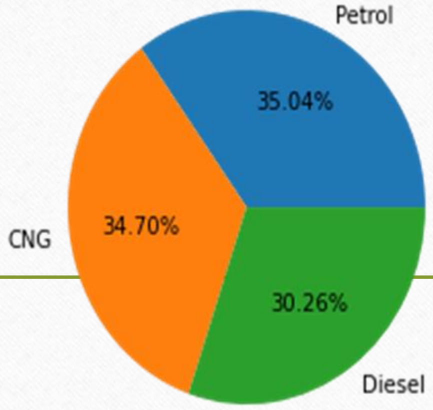
- The rating is very much dependent on the manufacturer.
- The cars from manufacturer 4 have the highest NCAP rating.



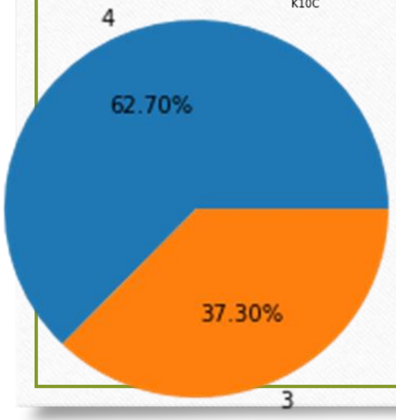
- Manufacturer 1 has the maximum number of cars whereas very less cars are from companies 2, 5, and 4.



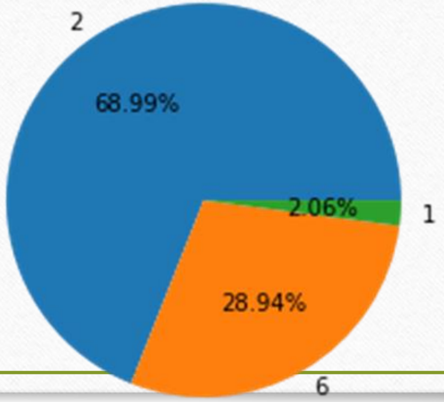
- F8D Petrol engine, 1.5 L U2 CRDi, and K Series Dual jet are the most commonly used engine types.



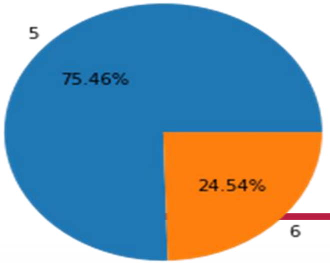
- Cars use nearly all types of fuel types but petrol is the maximum among them.



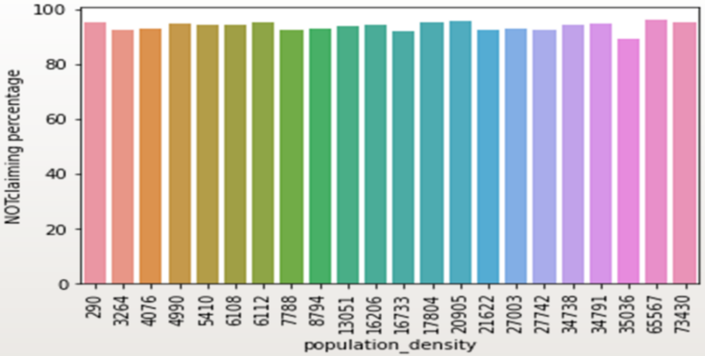
- Most cars have 4 cylinders present in the engine of the car.

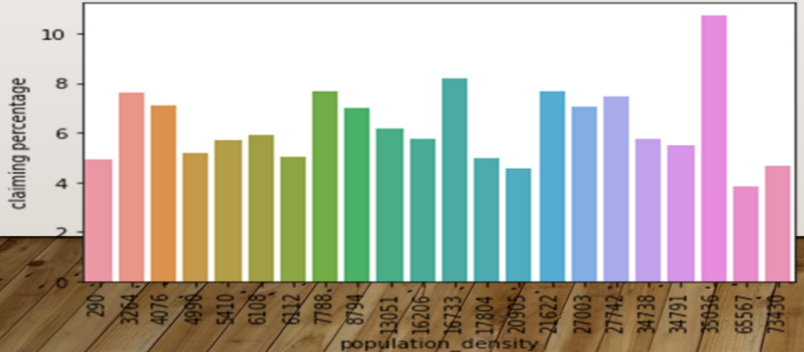


- Most of the cars have 2 airbags installed in the car.

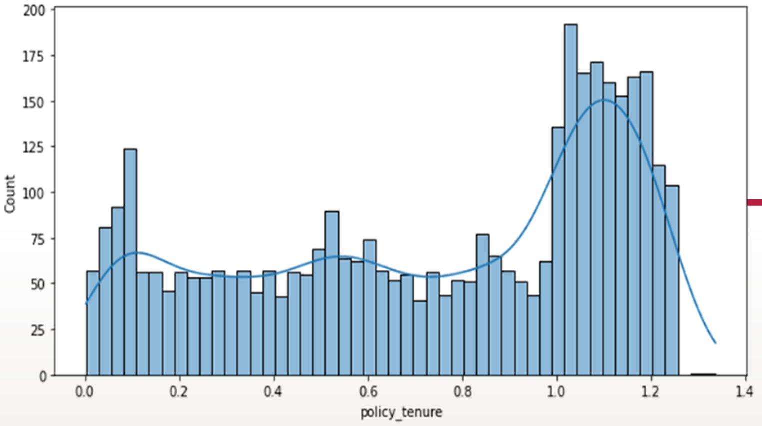


- Most of the cars have 5 gears installed in the car.

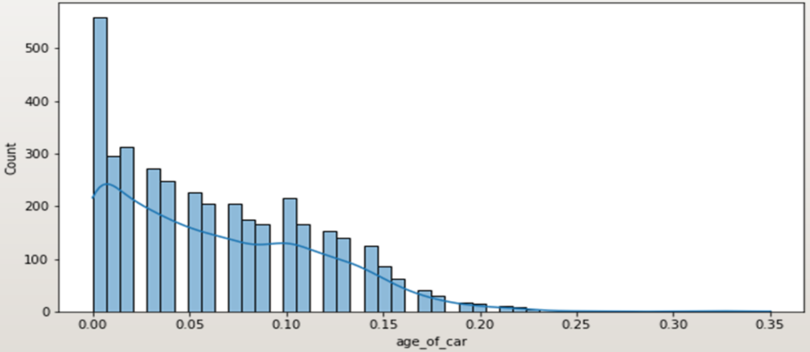




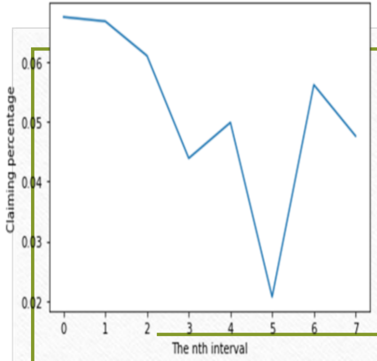
- population density does not have any significant effect on claiming insurance. It is mostly uniform throughout.



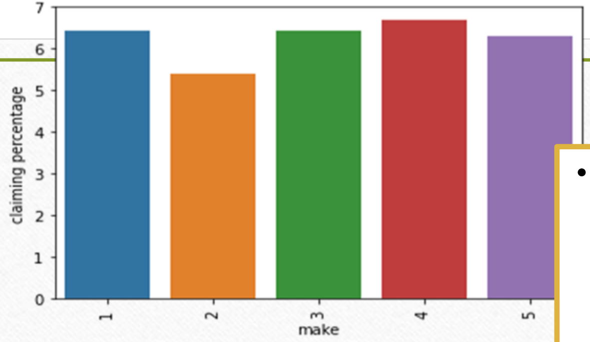
- In general, claiming is more for policies that have normalized tenure more than 1.



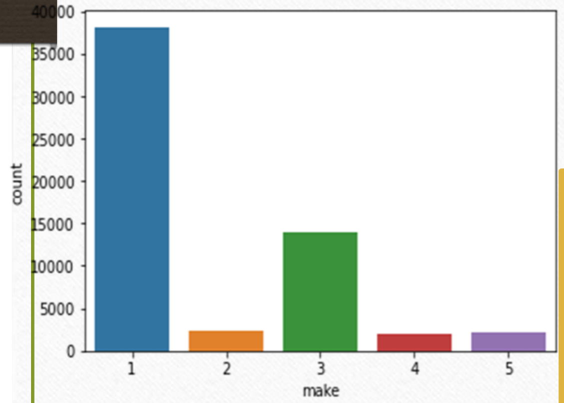
- Among the people going for the claim, the trend is that as the age of the car increases the number of claims decreases.



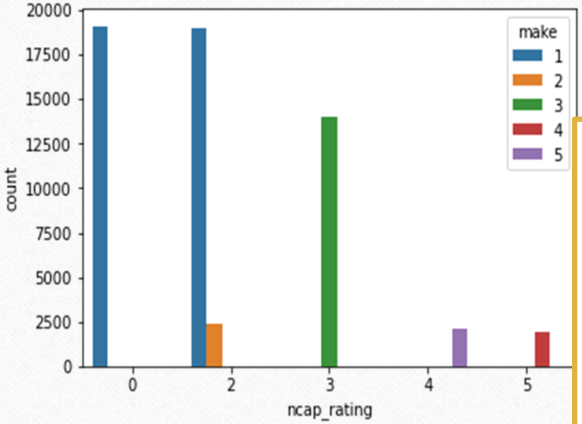
- In general, we can say that there is a gradual overall decrease in the percentage of claims as age increase.



- claiming percentage for manufacturer 2 is least when compared to others.

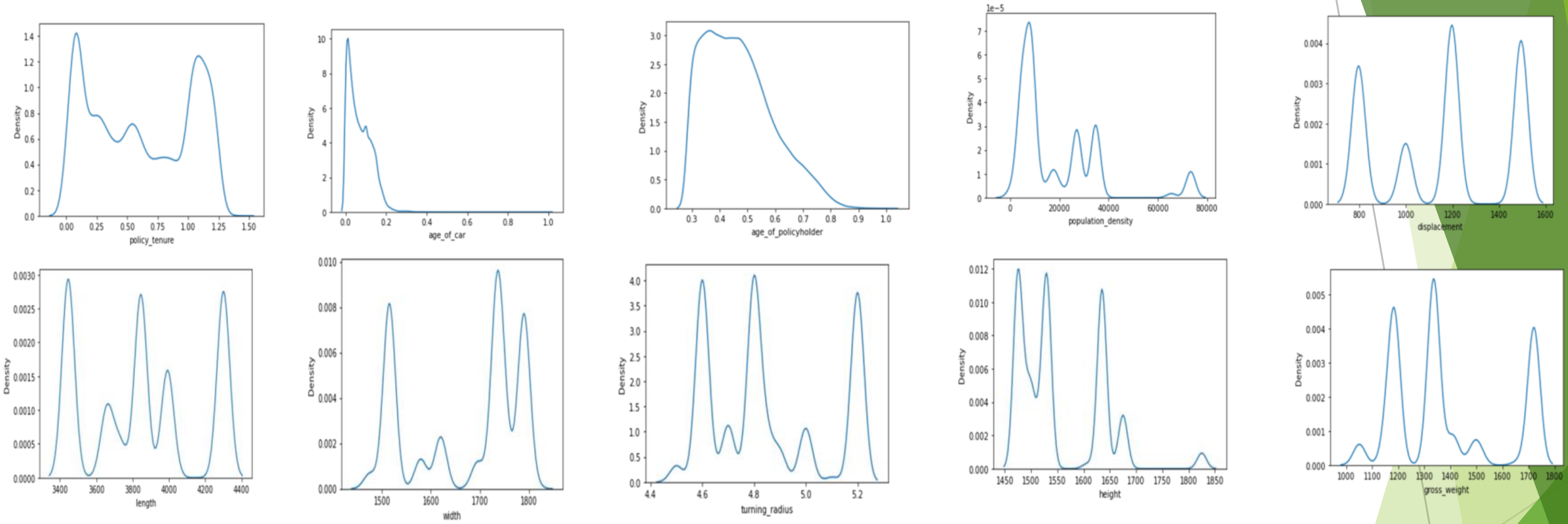


- the manufacturer 1 has most number of cars or customers in the given dataset followed by the manufacturer 3.



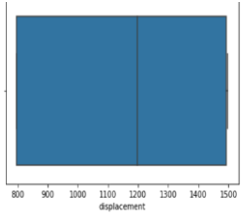
- the rating is very much dependent on the manufacturer. Manufacturer 1 has the lowest ratings, followed by 2, then 3,5, and 4 respectively.

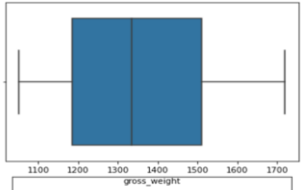
Distribution of Variables:

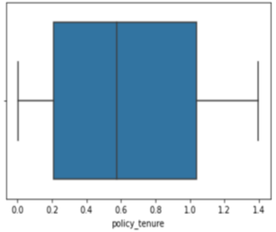


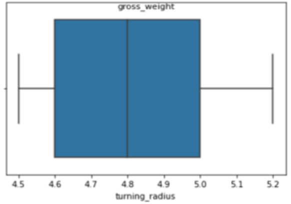
- The attributes are not normally distributed and are highly skewed.
- We will need to transform that distribution using transformation techniques like a central limit theorem, box cox, etc.
- We may also go for non-parametric tests as it does not need data to follow a specific distribution.

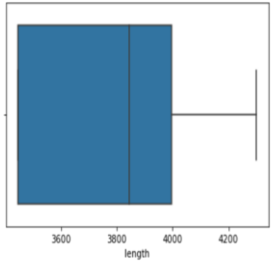
Outliers:

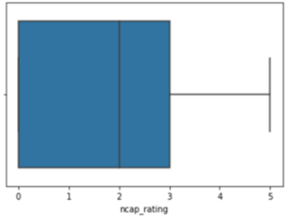


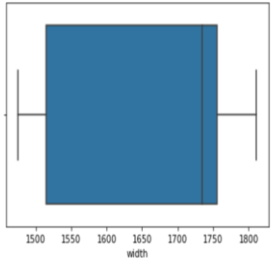


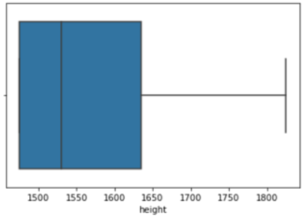






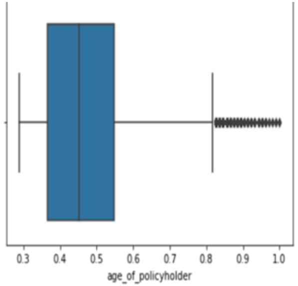


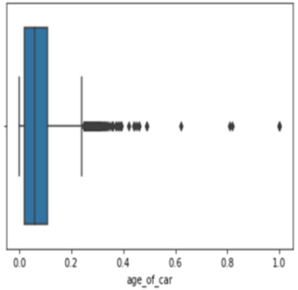


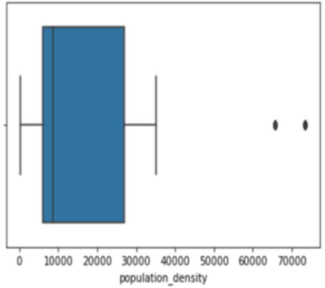


- age_of_car, age_of_policyholder and population_density have outliers present in them.

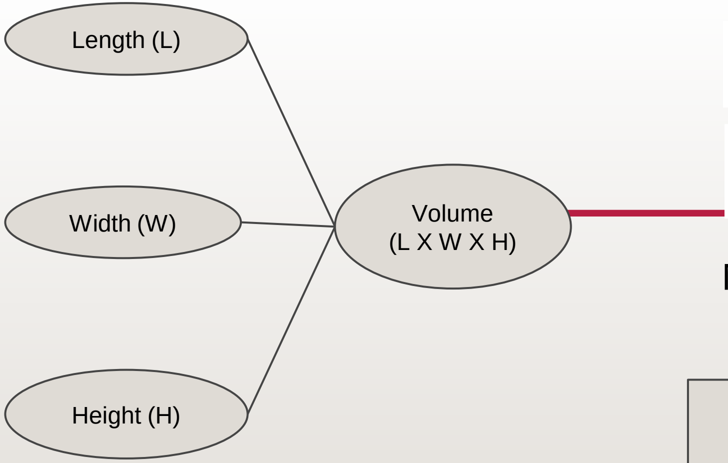
- We removed the outliers using the interquartile range by using different values for this depending on the number and position of outliers.







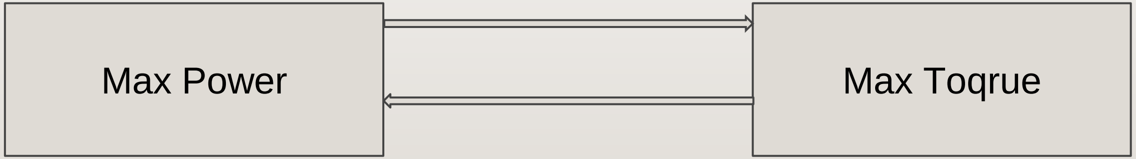
FEATURE ENGINEERING




```
Max Torque:  ['113Nm@4400rpm', '60Nm@3500rpm', '250Nm@2750rpm', '82.1Nm@3400rpm', '91Nm@4250rpm', '200Nm@1750rpm', '200Nm@3000rpm', '85Nm@3000rpm', '170Nm@4000rpm']
```

```
Max Power:  ['88.50bhp@6000rpm', '40.36bhp@6000rpm', '113.45bhp@4000rpm', '55.92bhp@5300rpm', '67.06bhp@5500rpm', '97.89bhp@3600rpm', '88.77bhp@4000rpm', '61.68bhp@6000rpm', '118.36bhp@5500rpm']
```

$$\text{MaximumTorque} = [P_{bhp} * 5252) / N_{rpm}] * 1.3558179483 \text{ nm}$$



$$\text{Maximum Power} = (\text{Torque}_{nm} * N_{rpm})/(5252 * 1.3558179483) \text{ bhp}$$

Feature Engineering Description:

We have developed three new feature engineering techniques by combining length , breadth and height into “volume” , derived max power and max torque using the scientific formulas, and found a combined score of all less insignificant columns into a single column called “feature_score”.

Finding the volume feature :

```
df_featured['volume']=df_featured['length'] * df_featured['width'] * df_featured['height']
df_featured.drop(['length','width','height'],axis=1,inplace=True)
```

Using the volume column of the dataset one can infer that the larger the volume of car is the bigger is the car specifying its segment making it fall into SUV category, similarly the lesser the volume of car making it fall into hatchback segment.

- **Calculation of max power to max torques:**

MaximumTorque = [(40.36bhp * 5252) / 6000rpm] * 1.3558179483 NewtonMeter

The function is performing a calculation to determine the maximum torque of an engine using the maximum power and the engine RPM.

- **Calculation of max torque to max power :**

BHP = (60 * 3500)/(5252 * 1.3558179483) bhp = 29.49 bhp

The function is performing a calculation to determine the maximum power of an engine using the maximum torque and the engine RPM. The calculation is based on the following formula: BHP = (60 * 3500)/(5252 * 1.3558179483)

BASE MODEL USING LOGISTIC REGRESSION

Completely imbalanced

	precision	recall	f1-score	support
0	0.94	1.00	0.97	16454
1	0.00	0.00	0.00	1124
accuracy			0.94	17578
macro avg	0.47	0.50	0.48	17578
weighted avg	0.88	0.94	0.91	17578

We can see that the recall value for our model is 0 for class 1 so our model is highly biased towards the majority class and so we will balance our data and than build the model.

Model Fine Tuning

- We built the same model with new data after SMOTE and verified the results
- We introduced SMOTE oversampling technique to increase the data dimension such that the target variable (is_claim) will be in the ratio 1:1.

```
xtrain1, xtest1, ytrain1, ytest1 = train_test_split(X2, y2, test_size=0.3,stratify=y2, random_state=48)
```

```
from sklearn.metrics import classification_report
model1=LR.fit(xtrain1,ytrain1)
ypred=model.predict(xtrain1)
print(classification_report(ytrain1,ypred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00     38390
           1       0.50      1.00      0.67     38391

    accuracy                           0.50     76781
   macro avg       0.25      0.50      0.33     76781
weighted avg       0.25      0.50      0.33     76781
```

Therefore balancing of target variable and better performance was achieved using SMOTE.

Decission Tree Classifier :

```
DT=DecisionTreeClassifier()
model2=DT.fit(xtrain4,ytrain4)
ypred=model2.predict(xtrain4)
ypred1=model2.predict(xtest4)
```

```
print(classification_report(ytrain4,ypred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     38468
           1       1.00      1.00      1.00     38313

    accuracy                           1.00     76781
   macro avg       1.00      1.00      1.00     76781
weighted avg       1.00      1.00      1.00     76781
```

```
print(classification_report(ytest4,ypred1))
```

```
              precision    recall  f1-score   support

           0       0.91      0.90      0.90     16376
           1       0.90      0.91      0.90     16531

    accuracy                           0.90     32907
   macro avg       0.90      0.90      0.90     32907
weighted avg       0.90      0.90      0.90     32907
```

Performance of the model is tremendously improved but the model is overfitted as the performance of train was better than test.

Random Forest Classifier:

```
RF=RandomForestClassifier()
model_RF=RF.fit(xtrain4,ytrain4)
ypred_RF=model_RF.predict(xtrain4)
ypred_RF=model_RF.predict(xtest4)
```

```
print(classification_report(ytest4,ypred_RF))
```

```
              precision    recall  f1-score   support

           0       0.87      0.88      0.88     16376
           1       0.88      0.87      0.87     16531

    accuracy                           0.87     32907
   macro avg       0.88      0.88      0.87     32907
weighted avg       0.88      0.87      0.87     32907
```

Its performance is comparitivery less than that of Decision tree and as the accuracies were marginally less when compared to overfitted Decision tree model.

Ada Boost Classifier:

```
AD = AdaBoostClassifier()
model_AD=AD.fit(xtrain4,ytrain4)
ypred_AD=model_AD.predict(xtrain4)
ypred_AD=model_AD.predict(xtest4)
```

```
print(classification_report(ytest4,ypred_AD))
```

```
              precision    recall  f1-score   support

           0       0.75      0.68      0.71     16376
           1       0.71      0.78      0.74     16531

    accuracy                           0.73     32907
   macro avg       0.73      0.73      0.73     32907
weighted avg       0.73      0.73      0.73     32907
```

Model performance is comparatively less than previous models.

Gradient Boosting Classifier:

```
GB = GradientBoostingClassifier()
model_GB=GB.fit(xtrain4,ytrain4)
ypred_GB=model_GB.predict(xtrain4)
ypred_GB1=model_GB.predict(xtest4)
```

```
print(classification_report(ytrain4,ypred_GB))
```

	precision	recall	f1-score	support
0	0.86	0.97	0.91	38468
1	0.96	0.84	0.90	38313
accuracy			0.91	76781
macro avg	0.91	0.91	0.91	76781
weighted avg	0.91	0.91	0.91	76781

```
print(classification_report(ytest4,ypred_GB1))
```

	precision	recall	f1-score	support
0	0.86	0.97	0.91	16376
1	0.97	0.84	0.90	16531
accuracy			0.91	32907
macro avg	0.91	0.91	0.91	32907
weighted avg	0.91	0.91	0.91	32907

Model performance is good and it is better than all previous models when it comes to generalization but the recall score in in 84% which suggested around 16% of False Negatives.

Extreme Gradient Boosting Classifier:

```
XGB = XGBClassifier()
model_XGB=XGB.fit(xtrain4,ytrain4)
ypred_XGB=model_XGB.predict(xtrain4)
ypred_XGB1=model_XGB.predict(xtest4)
```

```
print(classification_report(ytrain4,ypred_XGB))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     38468
           1       1.00      0.92      0.96     38313

    accuracy                           0.96     76781
   macro avg       0.96      0.96      0.96     76781
weighted avg       0.96      0.96      0.96     76781
```

```
print(classification_report(ytest4,ypred_XGB1))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     16376
           1       1.00      0.91      0.95     16531

    accuracy                           0.95     32907
   macro avg       0.96      0.95      0.95     32907
weighted avg       0.96      0.95      0.95     32907
```

The above model is comparatively better than all the previous models in terms of generalization and metrics.

It has better F1 score and recall when compared to Gradient Boosting which suggests the more precision in predicting our target variable.

CONCLUSION

- This model has a high level of accuracy with an overall accuracy score of 0.95. The precision scores for both classes are high, with class 0 having a precision score of 0.92 and class 1 having a precision score of 1.00. The recall score for class 0 is also high at 1.00, indicating that the model correctly identified all instances of class 0. The recall score for class 1 is slightly lower at 0.91, indicating that the model correctly identified 91% of instances of class 1.
- The F1-score is a measure that takes into account both precision and recall, and it indicates the overall performance of the model. The F1-score for both classes is high, with class 0 having an F1-score of 0.96 and class 1 having an F1-score of 0.95.
- Based on these metrics, it appears that the model is performing well for insurance claim prediction.

- **Final XGB Model can help the insurance company automate and streamline their claims processing operations, resulting in faster, more accurate, and cost-effective claims management. Predictive models can be used to estimate the likelihood of future claims and to determine the appropriate reserves and premiums to charge.**

THANK YOU