

# Machine learning - Hackathon

Vehicle Insurance Prediction

# IMPORT

- import numpy and pandas for numerical calculation and Data Handling & Processing.
- for Scaling , Encoding and Polinomial features for Hidden paterns: MinMaxScaler , OneHotEncoder , PolynomialFeatures , SelectKBest , f\_classif.
- for Imputing na : SimpleImputer
- Under and over sampling for imbalanced data set: RandomUnderSampler , RandomOverSampler
- For Dimentionality Reduction and as model: PCA
- test case split and Cross validation for right hyper parmeter : train\_test\_split , GridSearchCV
- Metrics to calculate Classification type target: accuracy\_score, classification\_report, confusion\_matrix , f1\_score , roc\_auc\_score.
- for Pipeline Creationa and transformimg : Pipeline , ColumnTransformer
- For chart and visuals : matplotlib.pyplot , Seaborn
- Joblib for saving the file
- Warnings to filterout unnecessary warninngs during Model Training.

# Data Loading and EDA process

- Importing data from using `pd.read_CSV` ( All train , test and Sample prediction data sets.)
- Using `df.info()` , `df.describe()` , `df.shape` , `df.head()` , `train_data.isna().sum()` , `train_data.duplicated().sum()` to get over view of data types , data size and no of columns and columns names , missing value and duplicated data informations are identified.
- Use of box plot , correlation matrix , scatterplot and pair plot for identifying the outliers and patterns and overall relationship and behaviour of data can be identified.
- Here we don't have any linear relationship and poor correlation are identified , and values are scattered overlapped and does not follow any standard pattern.
- Also unbalanced Target data is identified. We need to do either undersampling ( loss of features) or oversampling ( duplication of features). We will try both the methods.

# Preprocessing

- First we remove the Outlier , by using the comparison operator in train data.
- We planed to try models for both under sampled and over sampled one. First go with under sampling to train data.
- Different preprocessing was done for under sampling and oversampling ,
- Mapping values as bellow as shown for under sampling
- For over sampling i have done OHE to Vehicle age and Regional\_code is left as it is ( since during trying the oversampled data , we have already got good results in RF and tree based alorithms, since RF can handle regional\_code as it is , so we are planing to go with this .

## Undersampled\_Preprocessing:

```
data['Gender'] = data['Gender'].map({'Female': 0, 'Male': 1})
data['Vehicle_Damage'] = data['Vehicle_Damage'].map({'No': 0, 'Yes': 1})
data['Vehicle_Age'] = data['Vehicle_Age'].map({'< 1 Year': 0, '1-2 Year': 1 , '> 2 Years':3})
```

```
bins = [0, 10, 20, 30, 40, 52]
labels = ['1-10', '11-20', '21-30', '31-40', '41-52']
data['Region_group'] = pd.cut(data['Region_Code'], bins=bins, labels=labels)
data = data.drop(columns='Region_Code')
```

## Oversampled\_Preprocessing:

```
[15]: data['Gender'] = data['Gender'].map({'Female': 0, 'Male': 1})
      data['Vehicle_Damage'] = data['Vehicle_Damage'].map({'No': 0, 'Yes': 1})
```

```
[16]: data.head()
```

<div> ● ● ●

```
[17]: encoded_data = pd.get_dummies(data,columns = ['Vehicle_Age'],drop_first = True)
```

- Do all these steps by concat of train and test data to capture the column changes will use for testing.
- Then remove the 'id' column from the data.
- After all this split again the train and test data separately and save as Preprocessed train and preprocessed test data.
- NOTE : Due to Model performance time and One-hot encoded columns will be used for all models we have done this before the pipeline. We will be performing scaling separately in pipeline.
- Scaling should be performed to train data only, else lead to data leakage, that's the case for OHE.
- Before building pipeline do create a function for easy and fast calculation as below and split the data x and y based on target variable, and use train\_test\_split method to it.

```
def results(x_train , x_test , y_train , y_test , model):
    model.fit(x_train,y_train)
    train_predict = model.predict(x_train)
    test_predict = model.predict(x_test)
    print('Train F1_score:',f1_score(y_train , train_predict))
    print('Test F1_score:',f1_score(y_test , test_predict))
    print('Train Accuracy_score:\n',accuracy_score(y_train , train_predict))
    print('Test Accuracy_score:\n',accuracy_score(y_test , test_predict))
    print('Train confusion_matrix:\n',confusion_matrix(y_train , train_predict))
    print('Test confusion_matrix:\n',confusion_matrix(y_test , test_predict))
    print('Train Classification_Report:\n',classification_report(y_train , train_predict))
    print('Test Classification_Report:\n',classification_report(y_test , test_predict))
    print('Train score\n', roc_auc_score(y_train , train_predict))
    print('Test score\n', roc_auc_score(y_test , test_predict))
```

```
x = prepr_train.drop(columns=[ 'Response' ])
y = prepr_train[ 'Response' ]
```

```
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size= 0.3 , random_state= 42)
```

# Models Planned To Perform

- Logistic Regression ( Due to Binary target)
- Random Forest
- KNN
- Naive Bayes
- Boosting
- Voting
- Stacking

# Algorithms Performed

Under Sampled Data

- Linear Regression :
  1. **LR** (Poor Performance)
  2. **LR + Polynomial Features , degree = 2** .( to analysis unseen pattern in Linear model )
  3. **LR + Polynomial Features + Select K Best + PCA** ( to reduce and select important features and compressed to two using PCA.)
  4. → ( Since model have no or less linear relation ship , Performance is not Good)
- Random Forest :
  1. **RF with Scaled Data + Hyper parameters** . ( good compared to LR , since score is 73%).
  2. **GridsearchCV for previous model.** ( Same score not much difference in score)
  3. **RF with unscaled data + hyper parameters.**( Good performance compared to Scaled data, since score is in range of 83% )
- KNN:
  1. **KNN** –(Poor Performance like LR)
- Naive Bayes:
  1. **NB** –(Poor Performance like LR)

- Boosting :
  1. **ADA Boost with scaled data** ( performance is low even with estimator as DT).
  2. **ADA Boost with un-scaled data** ( Performance is Good as like RF we used before, score 82 %)
  3. **GridSearchCV with ADA Boost and DT parameter** ( No significant difference in score value as Previous score of 82 remains same)
  4. **XGB with unscaled data** ( apart all booting we tried untill now has good Score of 84% with minimal difference in train and test data.)



# Algorithms Performed

## Over Sampled Data

- Linear Regression :
  1. **LR** (Poor Performance)
  2. **LR + Polynomial Features** , **degree = 2** .( to analysis unseen pattern in Linear model )
  3. **LR + Polynomial Features + Select K Best + PCA** ( to reduce and select important features and compressed to two using PCA.)
  4. → ( Since model have no or less linear relation ship , Performance is not Good and results are as same as under sampled data. No improvement)
- Random Forest :
  1. **RF with unscaled data + Hyper parameters ( max\_depth = 5,15 ,20,35,45 and diffferent leaf and split )** .(Tried with different depth rate we got increaching accuracy untill 90% , Good mode).
- Boosting:
  1. **ADA Boosting with unscaled data with DT estimator:** ( Model performance is very Good slightly Seams to be Overfitting.)
  2. **XGB Boosting with unscaled data with DT estimator:** ( very promising Result and very Generalized Data accuracy up to 83% , both train and test as very little deviation.)
  3. **XGB Boosting with unscaled data with Hyper parameter :** ( no significan change in Output)

- KNN:
  1. **KNN** –(Poor Performance like LR)
- Naive Bayes:
  1. NB –(Poor Performance like LR)
- Voting:
  1. **Voting Soft** (Tried the combination of models that have predicted Good, ( Random forest and XG boost). ( Good results compared to all Models performed untill now) (4 models)
  2. **Voting hard**(Tried the combination of models that have predicted Good, ( Random forest and XG boost). No good compared to Soft Voting. We go with Soft Voting.
  3. **Voting Soft**( tried with 6 models , no good output compared with top 4 )
- Stacking:
  1. **Stacking with same models and final estimator as LR** ( performance is not but no as good as voting soft , Train and test difference is high)
  2. **Stacking with same models and final estimator as XGB** ( performance is not but no as good as voting soft )

# Creating Webapp and Model Deployment

- After model training use PKL to save using Joblib we have imported.
- After Saving we need to create the webapp, for that we use VS-Code to create. Create a file name of webapp.py , here we use streamlit to create a custom page set up.
- Here we need to test custom input , so we create every input columns need for prediction and the inputs go through the model and predict the output. Code file is attached with other document.
- To run the file use code “ **streamlit run weapp.py** “ check whether the models is working without any error.
- Now we are closer to deploy the model in GCP , for that we need Dockerfile and requirements.txt file, both are necessary to deploy model in a container image.
- In requirements.txt we need to specify the libraries need to perform the task and Dockerfile contains necessary information for creation a docker within GCP.
- Dockerfile consist of python version , working Directory , run commands to go through the requirements.txt and server address details and runcode that we use to run Web app.
- After Creating the all the required file , we use github as medium platform for Cloud , github platform easy to manage data changes compared to directly applying in GCP.
- In Github Create an repository for this project and save all the file ( PKL file , webapp.py , requirements.txt , train\_data.csv , Dockerfile ).

# Cloud Deployment

- After uploading in Github , Open the Google cloud platform ( many cloud platform available , but we use Google cloud ) . Select Create new project → console Run
- Select github from platform and login to you github platform , select allow all , then select Docker. Then click 'Create'.
- Select the accessability to Allow unathorized and give instance value as 1. than 'create'
- Model will automatically check and gives no error as below.
- After that copy the URL and can we shared. Using this anyone can access you model.

The screenshot displays the Google Cloud Run console for a service named 'vehicle-insurance-prediction'. The top navigation bar includes the Google Cloud logo, the service name, a search bar, and various utility icons. Below the navigation bar, there are links for 'Cloud Run', 'Service details', 'Edit & deploy new revision', and 'Edit Continuous Deployment'. The main content area shows the service's status as 'Ready', its region as 'us-central1', and its URL. A 'Revisions' tab is selected, showing a table of deployment revisions. The first revision, 'vehicle-insurance-prediction-00004-p6g', is the current one, with 100% traffic and deployed 4 hours ago. To the right of the table, a detailed view of the current revision is shown, including its name, deployment user, and various configuration settings like billing, concurrency, and autoscaling.

Google Cloud Vehicle insurance prediction run Search

Cloud Run Service details Edit & deploy new revision Edit Continuous Deployment Learn

vehicle-insurance-prediction Region: us-central1 URL: <https://vehicle-insurance-prediction-747732093596.us-central1.run.app> Scaling: Auto (Min: 1) Build

Metrics SLOs Logs Revisions Triggers Networking Security YAML

Revisions Manage traffic

Filter Filter revisions

Name	Traffic	Deployed ↓	Actions
vehicle-insurance-prediction-00004-p6g	100% (to latest)	4 hours ago	
vehicle-insurance-prediction-00003-g87	0%	4 hours ago	
vehicle-insurance-prediction-00002-sjs	0%	4 hours ago	
vehicle-insurance-prediction-00001-xcx	0%	4 hours ago	

vehicle-insurance-prediction-00004-p6g  
Deployed by san272thosh@gmail.com (logs, trigger) using Cloud Console

Containers Volumes Networking Security YAML

General

Billing	Request-based
Startup CPU boost	Enabled
Concurrency	80
Request timeout	300 seconds
Execution environment	Default

Autoscaling

Revision max instances	100
------------------------	-----

# Problems Faces

- Unable to upload the PLK file , since my file size neary 100 MB but github supports only max 25 Mb , compression take me hard time and also caused errors in GCP run.
- In option my i try to unload using Google Drive via link to maintain the orinality of the model , but not worked.
- Due to larger data size , unable to perform certain CV combinations , that took nearly more time than expected, Due to limited time i can up with this model.
- Due larger data size , GCP run didn't predict , even thought there is no error message , worked only when i increase the memory size in GCP configuration to 2GB.

Completed Model URL from GCP

<https://vehicle-insurance-prediction-747732093596.us-central1.run.app>

Analytics Vidhya Competition best Score:

The screenshot shows the Analytics Vidhya website for the 'Janatahack: Cross-sell Prediction' competition. The page has a dark theme. At the top, there's a navigation bar with the Analytics Vidhya logo, an 'Explore' dropdown, and a user profile icon 'S'. Below the navigation bar, the competition title 'Janatahack: Cross-sell Prediction' is prominently displayed. Underneath the title, there are tabs for 'About', 'Problem Statement', 'My Solution' (which is active), and 'Discuss'. The main content area is divided into two columns. The left column contains a table of solutions. The right column contains 'Registration Details' and 'Rounds' information. The 'Registration Details' section shows 21452 total registered users and 83 teams. The 'Rounds' section shows 'Round 1' is the current round. A 'Submit Solution' button is visible at the bottom of the right column. The table of solutions has columns for 'CODE FILE', 'SOLUTION FILE', 'PRIVATE SCORE', 'PUBLIC SCORE', and a status indicator. Two solutions are listed. The first solution has a private score of 0.7992333552 and a public score of 0.7957815235. The second solution has a private score of 0.8013204261 (highlighted in red) and a public score of 0.7967523925. Both solutions have a status indicator of a circle with a dot. The second solution's description is 'updated Sample and model softVoting\_OS\_1'.

Analytics Vidhya

Explore

# Janatahack: Cross-sell Prediction

About Problem Statement My Solution Discuss

CODE FILE	SOLUTION FILE	PRIVATE SCORE	PUBLIC SCORE	
N/A	Solution File	0.7992333552	0.7957815235	<input type="radio"/>
Solution Description: softVoting_OS_1_moremodels				
Submitted On 16 Mar 2025 • 08:05 AM				
N/A	Solution File	<u>0.8013204261</u>	0.7967523925	<input type="radio"/>
Solution Description: updated Sample and model softVoting_OS_1				

## Registration Details

21452  
Total registered

83  
Number of teams

## Rounds

**1** Round 1

Submit Solution