Theme Chosen : Strengthening Cybersecurity - Threat Detection

FRAUD DETECTION IN VEHICLE INSURANCE CLAIM

INTRODUCTION

The insurance industries consist of more than thousand companies in worldwide. And collect more than one trillions of dollars premiums in each year. The vehicle insurance fraud is the most prominent type of insurance fraud, which can be done by fake accident claim.

Also due to some flaws in the traditional process most of the companies are in search of some new technique to find fraud claims.

The insurance industry has recognized the importance of effective fraud management in recent times

In this project, focusing on detecting the vehicle fraud by using, machine learning techniques.

PROJECT OBJECTIVE

To build a classification methodology using machine learning techniques to determine whether a customer is placing a fraudulent insurance claim by using historical data.

Building the models and finding best suitable model for this application.

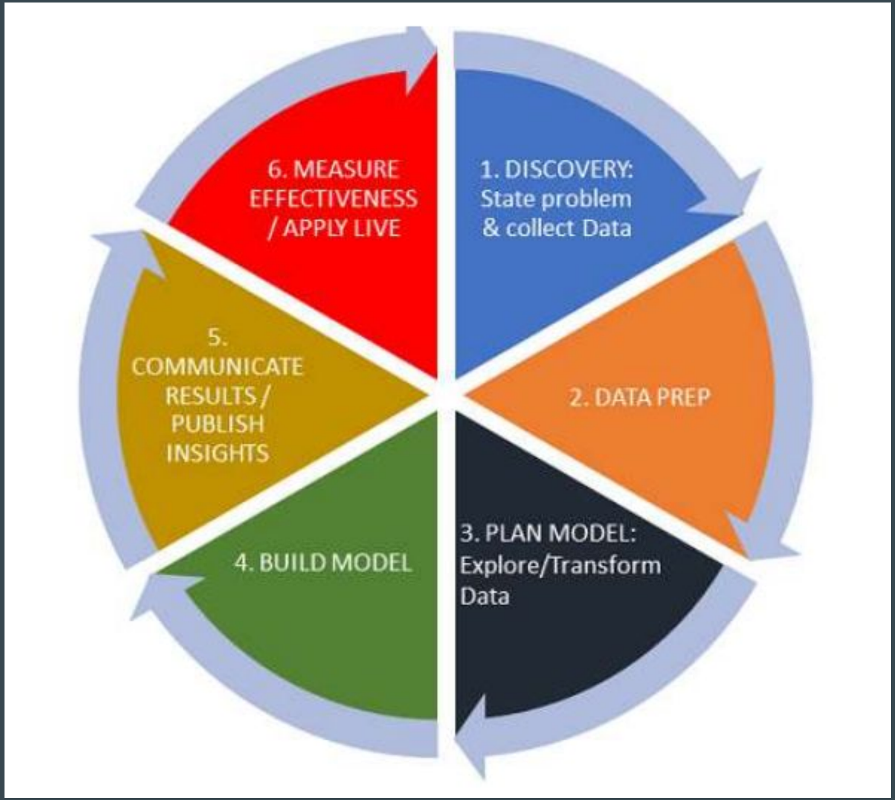
Comparative analysis of Machine Learning algorithms:

1. Random forest

2. Support Vector Classifier (SVC)

3. KNN (K Nearest Neighbour)

METHODOLOGY PROPOSED


1. DISCOVERY: State problem & collect Data
2. DATA PREP
3. PLAN MODEL: Explore/Transform Data
4. BUILD MODEL
5. COMMUNICATE RESULTS / PUBLISH INSIGHTS
6. MEASURE EFFECTIVENESS / APPLY LIVE

UNDERSTANDING THE PROBLEM

Step 1

Data Collection:

- Collected the dataset from Kaggle (Kaggle is an online community platform for data scientists and machine learning enthusiasts, allows us to use real time dataset that is collected by the professionals for future use)
- Dataset: insurance_claims.csv

Step 2

Data Preprocessing:

- Drop the columns not required for prediction.
- Handling Missing values - For this dataset, the null values ie. '?' have been replaced with NaN values. Check for null values in the columns. If present, impute the null values using the categorical imputer.
- Handling Categorical values - Replace and encode the categorical values with numeric values. For this One_Hot_Encoding of Scikit-learn is used.

Step 3

Model Building and Model Evaluation & Selection:

- Three machine learning models are built.
- By comparing three of them the final model for the vehicle fraud insurance claim detection is been selected.

RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique.

It follows the principle of ensemble learning, parallelization and is also a robust algorithm (robust to the outliers and noisy data).

It can be used for both Classification and Regression problems in machine learning.

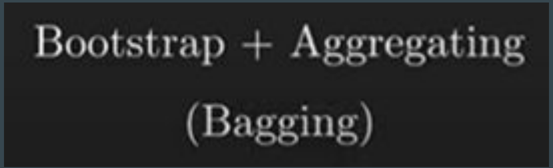
It takes less training time as compared to other algorithms.

It predicts output with high accuracy, even for the large dataset it runs efficiently.

It can also maintain accuracy when a large proportion of data is missing.

Two important steps while performing Random Forest Algorithm are:

1. Bootstrapping (create a new bootstrap dataset)
2. Aggregating

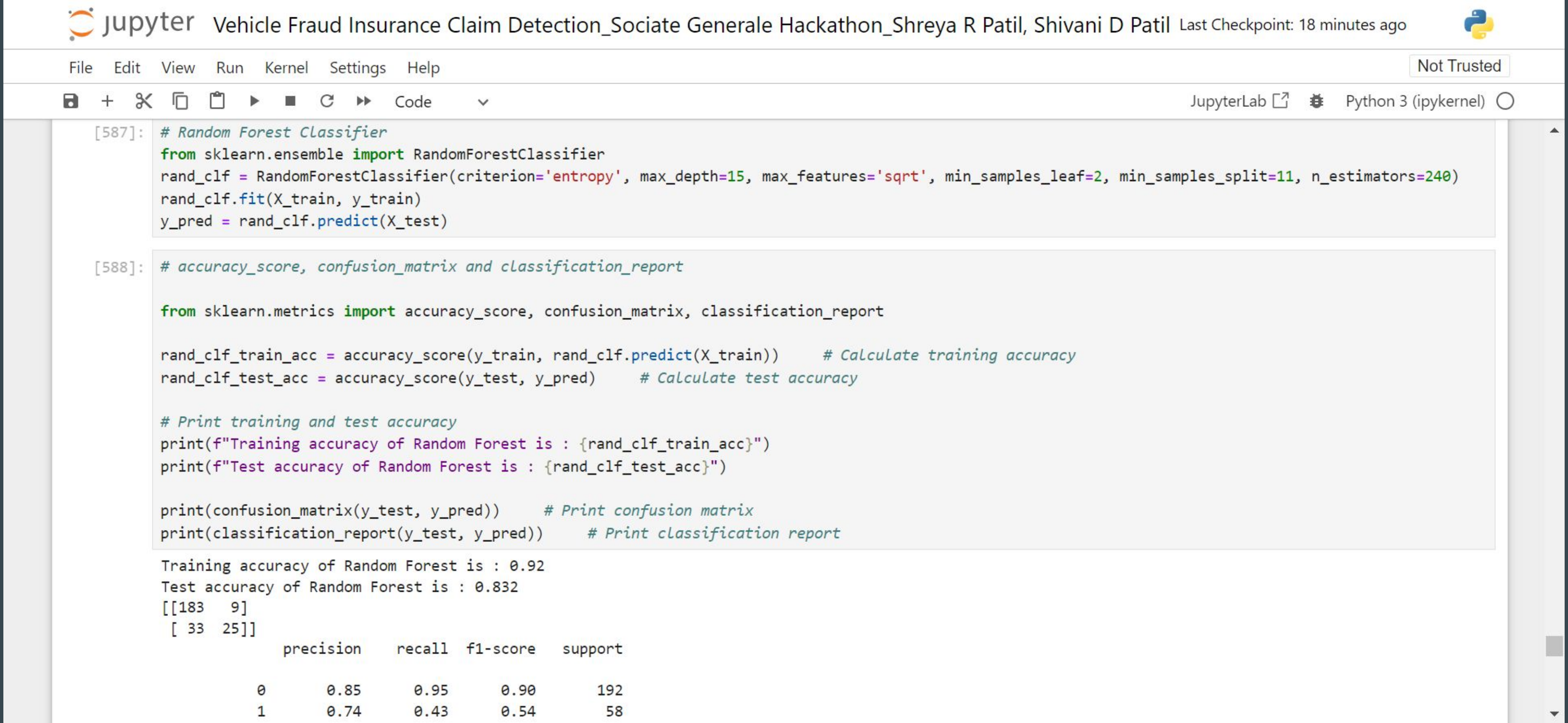


Bootstrap + Aggregating

(Bagging)

The combination of both of these steps is called as Bagging ie. (Bootstrapping + Aggregating)

RESULTS


jupyter
Vehicle Fraud Insurance Claim Detection_Sociate Generale Hackathon_Shreya R Patil, Shivani D Patil
Last Checkpoint: 18 minutes ago
File
Edit
View
Run
Kernel
Settings
Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[587]: # Random Forest Classifier
       from sklearn.ensemble import RandomForestClassifier
       rand_clf = RandomForestClassifier(criterion='entropy', max_depth=15, max_features='sqrt', min_samples_leaf=2, min_samples_split=11, n_estimators=240)
       rand_clf.fit(X_train, y_train)
       y_pred = rand_clf.predict(X_test)
```

```
[588]: # accuracy_score, confusion_matrix and classification_report

       from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

       rand_clf_train_acc = accuracy_score(y_train, rand_clf.predict(X_train))     # Calculate training accuracy
       rand_clf_test_acc = accuracy_score(y_test, y_pred)     # Calculate test accuracy

       # Print training and test accuracy
       print(f"Training accuracy of Random Forest is : {rand_clf_train_acc}")
       print(f"Test accuracy of Random Forest is : {rand_clf_test_acc}")

       print(confusion_matrix(y_test, y_pred))     # Print confusion matrix
       print(classification_report(y_test, y_pred))     # Print classification report
```

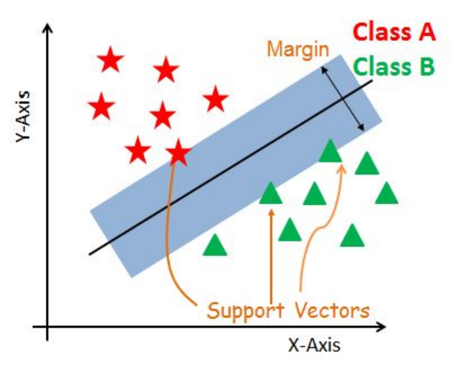
```
Training accuracy of Random Forest is : 0.92
Test accuracy of Random Forest is : 0.832
[[183   9]
 [ 33  25]]
              precision    recall  f1-score   support

           0       0.85      0.95      0.90       192
           1       0.74      0.43      0.54        58
```

SUPPORT VECTOR CLASSIFIER (SVC)

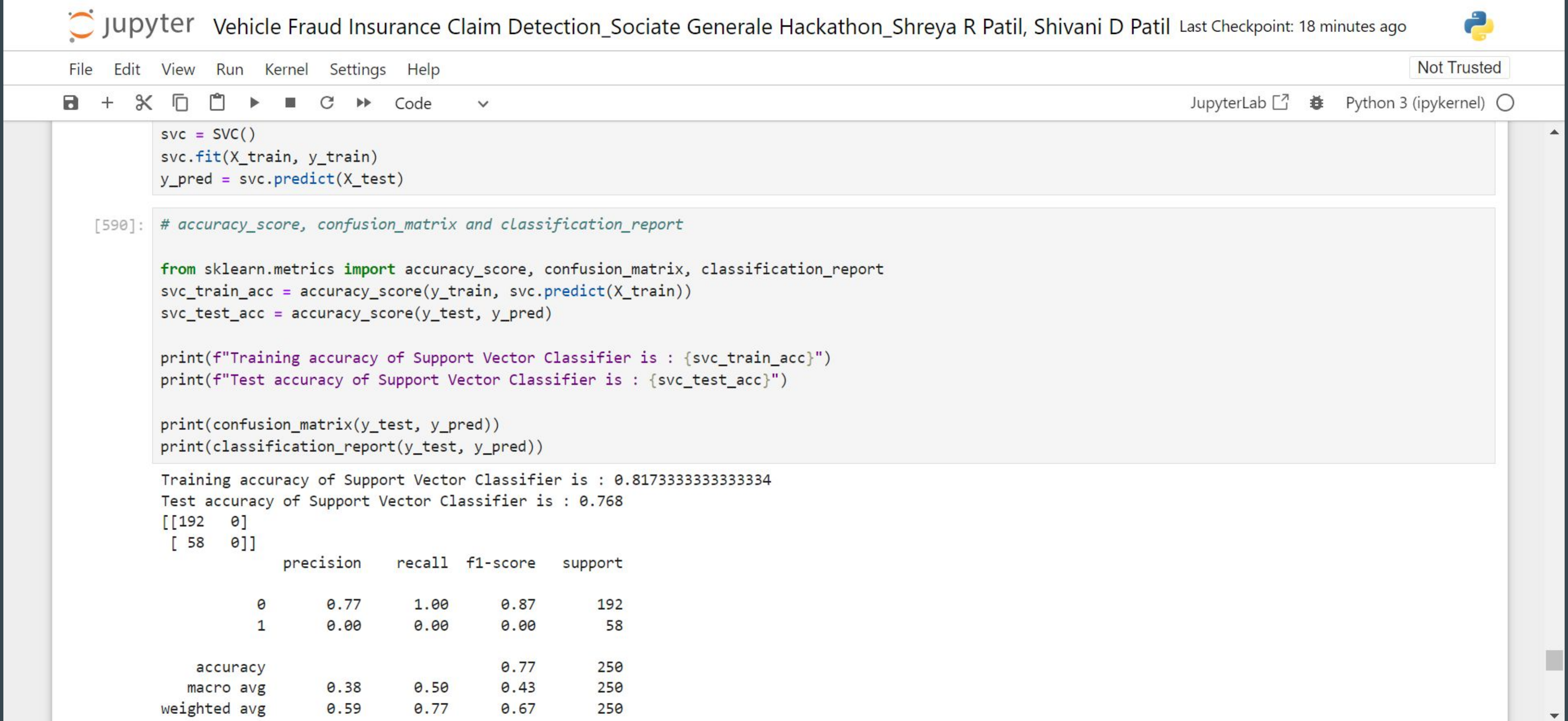
A Support Vector Classifier (SVC), also known as a Support Vector Machine (SVM) for classification, it is a linear model and is a type of supervised machine learning algorithm used for classification tasks (classifying the elements of a data set into two groups).

It belongs to a class of algorithms known as discriminative classifiers.

It is a binary classifier. (can classify the output into two types only) ie. in the case of vehicle fraud insurance claim detection it would classify as ‘Yes’ (indicating that yes the fraud is detected) or ‘No’ (indicating that no the fraud has not been detected).



RESULTS



jupyter
Vehicle Fraud Insurance Claim Detection_Sociate Generale Hackathon_Shreya R Patil, Shivani D Patil
Last Checkpoint: 18 minutes ago
File
Edit
View
Run
Kernel
Settings
Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)
[590]:

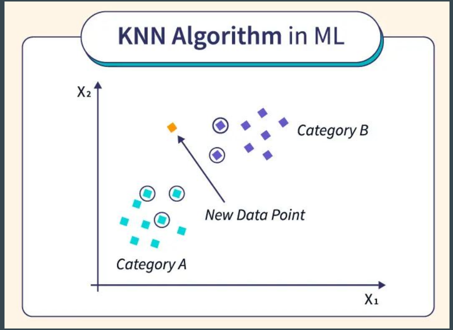
```
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
```

```
# accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
svc_train_acc = accuracy_score(y_train, svc.predict(X_train))
svc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Support Vector Classifier is : {svc_train_acc}")
print(f"Test accuracy of Support Vector Classifier is : {svc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of Support Vector Classifier is : 0.8173333333333334
Test accuracy of Support Vector Classifier is : 0.768
[[192   0]
 [ 58   0]]
              precision    recall  f1-score   support

           0       0.77      1.00      0.87       192
           1       0.00      0.00      0.00        58

    accuracy                           0.77       250
   macro avg       0.38      0.50      0.43       250
weighted avg       0.59      0.77      0.67       250
```

K - NEAREST NEIGHBOUR (KNN)

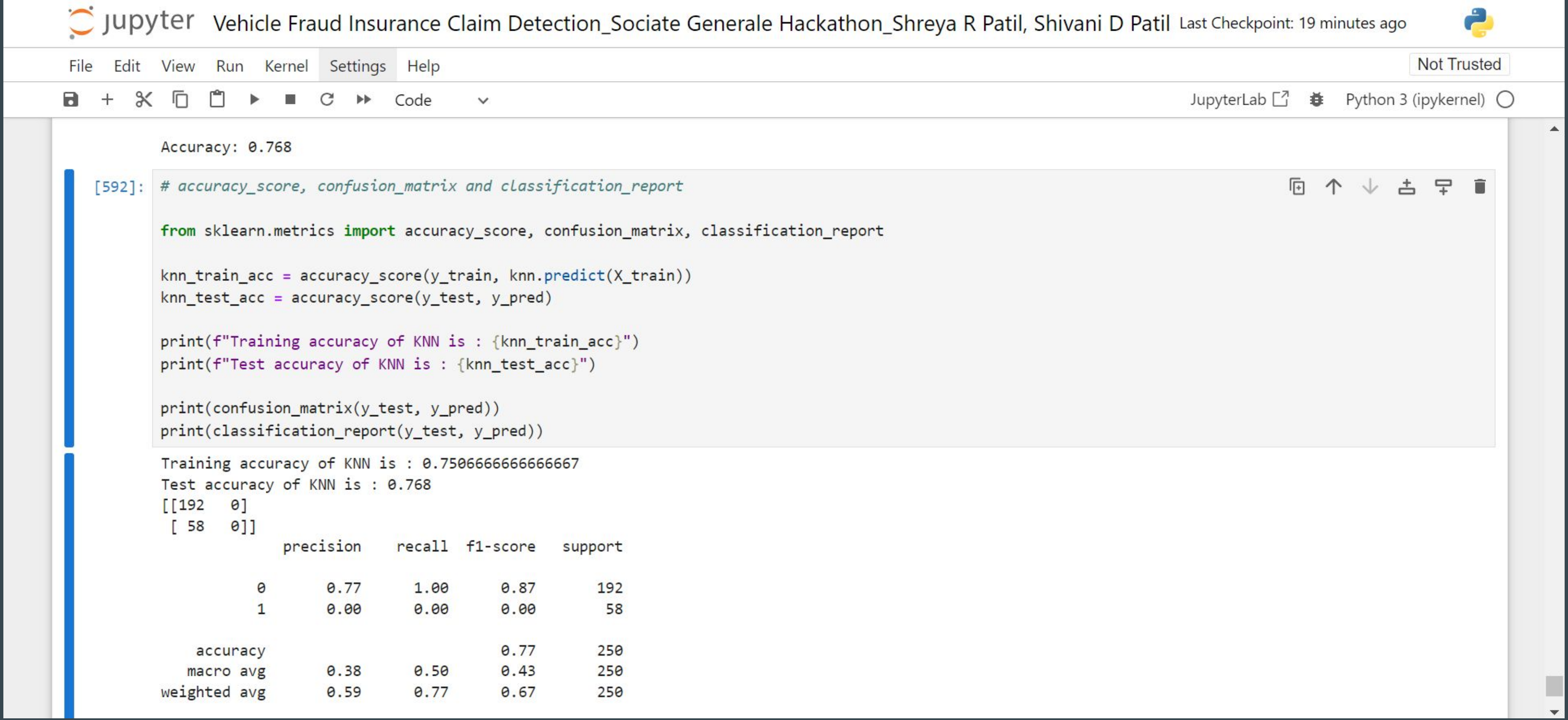
K-Nearest Neighbors (KNN) is a simple and widely used supervised machine learning algorithm for classification and regression tasks.



It is a non-parametric, instance-based learning algorithm, meaning that it makes predictions based on the similarity of data points in the training dataset.
It is a lazy learner (it doesn't build an explicit model during the training phase, instead, it stores the entire training dataset in memory and makes predictions at the time of inference).

KNN makes predictions by measuring the distance (e.g., Euclidean distance) between the input data point and the neighboring data points in the feature space.

RESULTS



jupyter
Vehicle Fraud Insurance Claim Detection_Sociate Generale Hackathon_Shreya R Patil, Shivani D Patil
Last Checkpoint: 19 minutes ago
File
Edit
View
Run
Kernel
Settings
Help
Not Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
Accuracy: 0.768
```

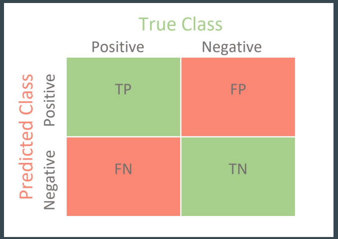
[592]:

```
# accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

knn_train_acc = accuracy_score(y_train, knn.predict(X_train))
knn_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of KNN is : {knn_train_acc}")
print(f"Test accuracy of KNN is : {knn_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of KNN is : 0.7506666666666667
Test accuracy of KNN is : 0.768
[[192   0]
 [ 58   0]]
              precision    recall  f1-score   support

           0       0.77      1.00      0.87       192
           1       0.00      0.00      0.00        58

    accuracy                           0.77       250
   macro avg       0.38      0.50      0.43       250
weighted avg       0.59      0.77      0.67       250
```

Evaluation Metric

CONFUSION MATRIX



		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

True Positive (TP)

- Model predicted the positive class correctly, and the true class is also positive.

True Negative (TN)

- Model predicted the negative class correctly, and the true class is also negative.

False Positive (FP)

- Model predicted the positive class incorrectly, and the true class is actually negative.

False Negative (FN)

- Model predicted the negative class incorrectly, and the true class is actually positive.

COMPARING MACHINE LEARNING MODELS

↓ Evaluation Parameters	Random Forest Classifier	SVC (Support Vector Classifier)	KNN (K - Nearest Neighbour)
Accuracy	83.2%	76.8%	76.8%
False Positive (FP) Values (from Confusion Matrix)	33	58	58
Prediction Speed	Moderate	Fast	Depends
Average Prediction Accuracy	Higher	Lower	Lower
Training speed	Fast	Fast	Fast (Excluding Feature extraction)

DESCRIPTION BY COMPARING THE MODELS

Thus, from the above table we can conclude that the Random Forest Classifier would be the most optimized algorithm to detect the fraud for vehicle insurance claim detection. Because, it has the highest accuracy among the all three algorithms and also the value for False Positive parameter is less for this algorithm which indicates it is the most precise algorithm.
Also the average prediction accuracy for the random forest classifier is higher.

Importance of FP parameter: it indicates the number for the cases where the fraud is actual happened but the model is not being able to predict that fraud. (So, such number should be minimum)

FUTURE ENHANCEMENT

Fraud detection system should be able to process the custom dataset uploaded at the front-end phase of the system, with visual graphs and remarks, and detect fraud automatically.

A front-end model can be built which would take the input from the user side and would then predict the output and classify it as Yes or No.

Also we can explore more by using the deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for feature extraction and fraud detection. These approaches would help to capture complex patterns in image, text, and time-series data.

Enhancement on the model transparency and interpretability can be done to build trust among users and stakeholders. XAI techniques can help provide explanations for model predictions and highlight important features contributing to fraud detection.

WEB APPLICATION USING STREAMLIT

Web application can be built using streamlit library in python.
There we will be creating 3 python files, namely:

1. app.py

   This will be the actual execution page, where we import other two file functions, show_predict_page() and show_explore_page(). This allows us to make a user interface tabs for prediction of fraud and exploring the dataset(data Visualization)

2. Predict.py

   This will be the file where our prediction model is get load using joblib and dataset is get load using pickel. This same file will consist of input tabs, that we ask from users. Input tabs can be in the form of selectbox, slider, textInputs, integerInputs depending upon data columns.

   The page will also have predict button which will display the fraud or genuine insurance claims according to the conditions are satisfied by the user inputs.

3. Explore.py

   All the visualization graphs, plots would be plotted here.

THANK YOU