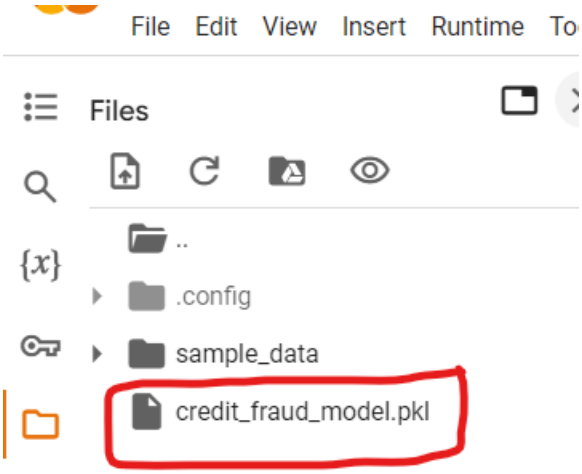
Setting up and running the data science project

How to RUN pkl file

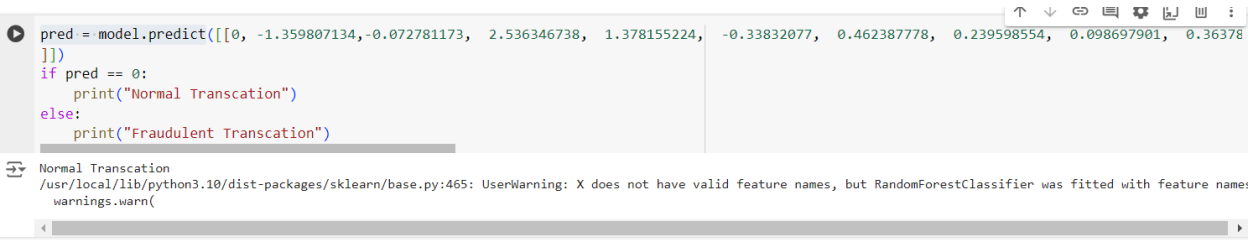
- Open any workbook example Google colab
- Mount drive and upload the pkl file



- In work space Import joblib
- Open the pkl file in joblib

```
[2] import joblib
    model=joblib.load('credit_fraud_model.pkl')
```

- In pred = model.predict give the transaction values which user need o find the normal or fraud transation
- Run the script



```
pred = model.predict([[0, -1.359807134,-0.072781173,  2.536346738,  1.378155224,  -0.33832077,  0.462387778,  0.239598554,  0.098697901,  0.36378
]])
if pred == 0:
    print("Normal Transcation")
else:
    print("Fraudulent Transcation")
```

```
Normal Transcation
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature name
  warnings.warn(
```

Purpose

Credit card fraud detection is a critical challenge in financial security, given the increasing incidence of fraudulent transactions and the consequential financial losses. This project addresses this challenge by developing a machine learning model to detect fraudulent credit card transactions using a dataset of transactions from European cardholders. The dataset, characterized by a significant class imbalance with only 0.172% of transactions being fraudulent, necessitates advanced techniques for effective fraud detection. Our approach includes exploratory data analysis (EDA) to understand the data, data preprocessing to handle missing values and feature scaling, and resampling techniques such as SMOTE to address class imbalance. We employ a RandomForestClassifier, known for its robustness in handling imbalanced data, and optimize it through hyperparameter tuning. The model's performance is evaluated using precision, recall, F1-score, and ROC-AUC metrics. The final model is deployed with provisions for real-time processing and continuous improvement based on feedback. This work aims to enhance fraud detection accuracy while ensuring scalability and interpretability.

Problem Statement

A credit card is one of the most used financial products to make online purchases and payments. Though the Credit cards can be a convenient way to manage your finances, they can also be risky. Credit card fraud is the unauthorized use of someone else's credit card or credit card information to make purchases or withdraw cash.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

We have to build a classification model to predict whether a transaction is fraudulent or not.

Data Souce

Kaggle

Proposed Methodology

1. Data Preprocessing:

- **Exploratory Data Analysis (EDA):** Analyze the dataset to understand its structure and identify patterns. Visualize class distribution to confirm the imbalance.
- **Handling Missing Values:** Impute missing values using appropriate methods or remove rows/columns with excessive missing data.
- **Feature Scaling:** Standardize features using techniques like normalization or log transformation to ensure all features contribute equally to the model.

2. Addressing Class Imbalance:

- **Resampling Techniques:** Use SMOTE to generate synthetic samples for the minority class (fraud) to balance the dataset. This helps the model learn from a more representative sample of fraudulent transactions.
- **Cost-Sensitive Learning:** Adjust class weights in the model to penalize misclassifications of the minority class more heavily, improving fraud detection sensitivity.

3. Model Selection and Training:

- **Algorithm Choice:** Employ a RandomForestClassifier due to its robustness and ability to handle imbalanced datasets effectively.
- **Hyperparameter Tuning:** Optimize the RandomForestClassifier using Grid Search to find the best combination of parameters (e.g., n_estimators, max_depth) that improves model performance.

4. Model Evaluation:

- **Performance Metrics:** Evaluate the model using precision, recall, F1-score, and ROC-AUC to measure its ability to correctly classify fraudulent transactions while minimizing false positives and false negatives.
- **Cross-Validation:** Use stratified k-fold cross-validation to assess the model's performance and ensure it generalizes well to unseen data.

5. Deployment and Monitoring:

- **Real-Time Processing:** Implement the model in a real-time environment to handle live transaction data. Develop integration with existing fraud prevention systems.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
confusion_matrix
from imblearn.over_sampling import SMOTE

# Load the dataset
data = pd.read_csv('/content/drive/MyDrive/DATA
SCIENCE/datasets/credit_card.csv')

# Exploratory Data Analysis (EDA)
print("Data Overview:")
print(data.head())  # Display first few rows
print("\nData Summary:")
print(data.describe())  # Summary statistics
print("\nData Info:")
print(data.info())  # Information about data types and missing
values

# Plot distribution of target variable
plt.figure(figsize=(6, 4))
sns.countplot(x='Class', data=data)
plt.title('Distribution of Fraudulent vs Non-Fraudulent
Transactions')
plt.show()
```

```
# Data Cleaning
# Handle missing values if any (uncomment if needed)
# data = data.fillna(method='ffill')

# Feature Engineering
# Apply logarithmic transformation to 'Amount' feature
data['Amount'] = np.log1p(data['Amount'])

# Dealing with Imbalanced Data
X = data.drop('Class', axis=1)  # Features
y = data['Class']  # Target variable

# Apply SMOTE to balance the data
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_resampled,
y_resampled, test_size=0.3, random_state=42)

# Model Selection
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Model Validation
y_pred = model.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Hyperparameter Tuning
param_grid = {
    'n_estimators': [100, 200],  # Number of trees
    'max_depth': [10, 20]  # Maximum depth of trees
}

grid_search = GridSearchCV(estimator=model,
param_grid=param_grid, cv=3, scoring='f1')
grid_search.fit(X_train, y_train)
print("\nBest Parameters from Grid Search:")
print(grid_search.best_params_)
```

```
# Optional: Retrain the model with the best parameters if needed
best_model = grid_search.best_estimator_
y_pred_best = best_model.predict(X_test)
print("\nClassification Report for Best Model:")
print(classification_report(y_test, y_pred_best))
import joblib

# Save the best model
joblib.dump(best_model, 'credit_fraud_model.pkl')

model=joblib.load('credit_fraud_model.pkl')
```

```
pred = model.predict([[0, -1.359807134,-
0.072781173,  2.536346738,  1.378155224,  -
0.33832077,  0.462387778,  0.239598554,  0.098697901,  0.3637869
7, 0.090794172,  -0.551599533, -0.617800856,-0.991389847,  -
0.311169354, 1.468176972 ,-0.470400525,0.207971242,  0.02579058,
0.40399296,0.251412098, -0.018306778, 0.277837576,-
0.11047391,  0.066928075,0.128539358 ,-0.189114844
,0.133558377  ,-0.021053053 ,149.62
]])
if pred == 0:
    print("Normal Transcation")
else:
    print("Fraudulent Transcation")
```