```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd

df = pd.read_parquet('/content/drive/MyDrive/DS4A_dataset/credit_card_data_da.parquet', engine='pyarrow')
df.to_csv('credit_card_data_da.csv')
```

```
df.info()
```

```
 7   Use Chip                         object
 8   Merchant Name                    int64
 9   Merchant City                    object
 10  Merchant State                   object
 11  Zip                              float64
 12  MCC                              int64
 13  Errors?                          object
 14  Is Fraud?                        object
 15  hour                             object
 16  minute                           object
 17  date                             datetime64[ns]
 18  datetime                         datetime64[ns]
 19  time_of_day                      object
 20  target                           int64
 21  charge_off                       float64
 22  merchant_city_rome               bool
 23  Person                           object
 24  Current Age                      int64
 25  Retirement Age                   int64
 26  Birth Year                       int64
 27  Birth Month                      int64
 28  Gender                           object
 29  Address                          object
 30  Apartment                        float64
 31  City                             object
 32  State                            object
 33  Zipcode                          int64
 34  Latitude                         float64
 35  Longitude                        float64
 36  Per Capita Income - Zipcode      float64
 37  Yearly Income - Person           float64
 38  Total Debt                       float64
 39  FICO Score                       int64
 40  Num Credit Cards                 int64
 41  personal_to_zipcode_income_diff  float64
 42  total_debt_personal_income_ratio float64
 43  total_debt_cards_ratio           float64
 44  CARD INDEX                       int64
 45  Card Brand                       object
 46  Card Type                        object
 47  Card Number                      int64
 48  Expires                          object
 49  CVV                              int64
 50  Has Chip                         object
 51  Cards Issued                     int64
 52  Credit Limit                     float64
 53  Acct Open Date                   object
 54  Year PIN last Changed            int64
 55  Card on Dark Web                 object
 56  level_2                          int64
 57  rolling_charge_off               float64
 58  rolling_fraud_count              float64
 59  rolling_tran_count               float64
 60  rolling_tran_volume              float64
 61  transaction_count                float64
 62  years_since_pin_change           int64
dtypes: bool(1), datetime64[ns](2), float64(18), int64(22), object(20)
memory usage: 3.2+ GB
```

```
df.head(10)
```

	User	Card	Year	Month	Day	Time	Amount	Use Chip	Merchant Name	Merchant City	...	Acct Open Date	Year PIN last Changed	Card on Dark Web	level_2	rolling_c
0	0	0	2016	1	3	10:48	66.48	Chip Transaction	-3345936507911876459	La Verne	...	09/2002	2008	No	0	
1	0	0	2016	1	4	06:43	40.02	Chip Transaction	-34551508091458520	La Verne	...	09/2002	2008	No	1	
2	0	0	2016	1	7	09:30	54.11	Chip Transaction	4055257078481058705	La Verne	...	09/2002	2008	No	2	
3	0	0	2016	1	7	16:03	89.48	Chip Transaction	3414527459579106770	Monterey Park	...	09/2002	2008	No	2	
4	0	0	2016	1	10	06:38	29.15	Chip Transaction	-5475680618560174533	Monterey Park	...	09/2002	2008	No	3	
5	0	0	2016	1	13	06:37	120.00	Chip Transaction	-4282466774399734331	Mira Loma	...	09/2002	2008	No	4	
6	0	0	2016	1	13	13:52	56.87	Chip Transaction	3527213246127876953	La Verne	...	09/2002	2008	No	4	
7	0	0	2016	1	15	10:56	1.44	Chip Transaction	-7232193519160172381	La Verne	...	09/2002	2008	No	5	
8	0	0	2016	1	18	16:57	102.90	Online Transaction	208649686760524778	ONLINE	...	09/2002	2008	No	6	

```
df.isnull().sum()
```

```
User                      0
Card                      0
Year                      0
Month                     0
Day                       0
                        ...
rolling_fraud_count    8469
rolling_tran_count     8469
rolling_tran_volume    8469
transaction_count      8469
years_since_pin_change    0
Length: 63, dtype: int64
```

```
df["Is Fraud?"].value_counts()
```

```
No     6869425
Yes       8412
Name: Is Fraud?, dtype: int64
```

```
# Subset specific columns
columns_to_select = ['Year', 'Day', 'hour', 'Amount', 'Use Chip', 'Merchant Name', 'MCC', 'Is Fraud?']
df = df[columns_to_select]
```

```
# Replacing values in the "Is Fraud?" column
df['Is Fraud?'] = df['Is Fraud?'].map({'No': 0, 'Yes': 1})
```

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 1.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.23.5)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.3)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.0)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.5.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2023.3.
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_enc
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encoders)
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

```
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import FunctionTransformer, LabelEncoder, OneHotEncoder
from imblearn.under_sampling import RandomUnderSampler
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
from category_encoders.binary import BinaryEncoder
```

```
from sklearn.preprocessing import StandardScaler
import category_encoders as ce

def clean(df):
    # Convert data type
    df['hour'] = df['hour'].astype('float')

    # Scale the "Amount" column
    scaler = StandardScaler()
    df['Amount'] = scaler.fit_transform(df[['Amount']])

    # Binary encoding for categorical variables
    cat_col = ['Use Chip', 'Day']
    for col in cat_col:
        if col in df.columns:
            be = ce.BinaryEncoder(drop_invariant=False)
            enc_df = pd.DataFrame(be.fit_transform(df[col]), dtype='int8')
            df = pd.concat([df, enc_df], axis=1)
            df.drop([col], axis=1, inplace=True)

    for col in df.columns:
        df[col] = df[col].astype(float)

    return df

# Create the pipeline
preprocessing_pipeline = Pipeline([
    ('cleaning', FunctionTransformer(clean, validate=False)),
], verbose=True)

df_transformed = preprocessing_pipeline.fit_transform(df)
```

```
Warning: No categorical columns found. Calling 'transform' will only return input data.
[Pipeline] .......... (step 1 of 1) Processing cleaning, total=  15.7s
```

Under sampling

Due to limitation of computational capacity, I subset 40000 data with 20% of them being fraud cases in order to balance the proportion and ensure model performance.

```
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split

# Split the dataset into features (X) and target variable (y)
X = df_transformed.drop(columns=['Is Fraud?'])
y = df_transformed['Is Fraud?']

# Calculate the desired number of fraud cases based on the desired proportion
desired_proportion = 0.2
total_samples = 40000
fraud_samples = int(total_samples * desired_proportion)

# Create RandomUnderSampler with the desired sampling strategy
rus = RandomUnderSampler(sampling_strategy={0: total_samples - fraud_samples, 1: fraud_samples}, random_state=1613)

# Apply random undersampling to the original dataset
X_resampled, y_resampled = rus.fit_resample(X, y)

# Split the resampled data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=1613)
```

Predictive Modeling with Random Forest

```
# Modeling with Random Forest
from sklearn.ensemble import RandomForestClassifier

rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)

y_pred_rf = rf_classifier.predict(X_test)

print("Random Forest Classifier Results:")
print(classification_report(y_test, y_pred_rf))
print(confusion_matrix(y_test, y_pred_rf))
```

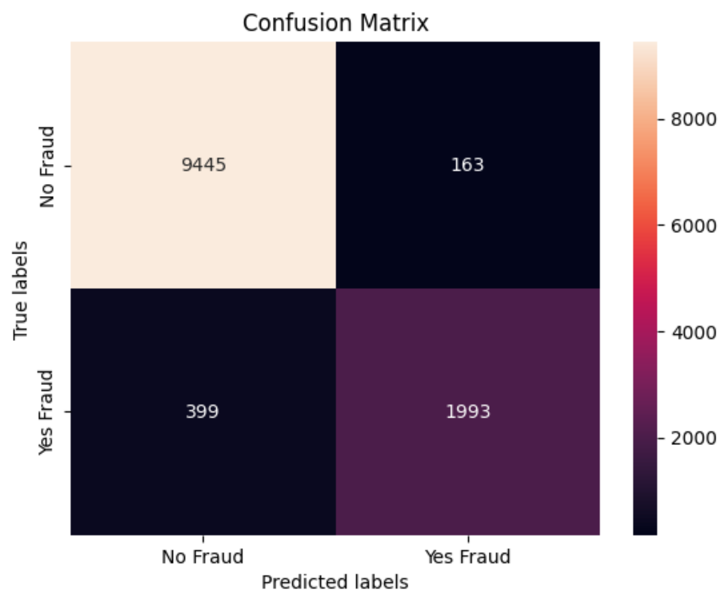
```
Random Forest Classifier Results:
              precision    recall  f1-score   support

         0.0       0.96      0.98      0.97      9608
         1.0       0.91      0.83      0.87      2392

    accuracy                           0.95     12000
   macro avg       0.94      0.91      0.92     12000
weighted avg       0.95      0.95      0.95     12000

[[9417  191]
 [ 404 1988]]
```

```
# Hyperparameters Tuning
import warnings
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Suppress all warnings
warnings.simplefilter("ignore")

# Define the hyperparameters
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'max_features': ['sqrt', 'log2'],  # Removed 'auto' and kept 'sqrt'
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Create a RandomForestClassifier model
rf = RandomForestClassifier(random_state=42)

# GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
                           cv=3, n_jobs=-1, verbose=0, scoring='f1_macro')

grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best hyperparameters:", best_params)

# Use the best estimator for predictions or further work
best_rf = grid_search.best_estimator_

y_pred_best_rf = best_rf.predict(X_test)

print("Random Forest Classifier Results with Best Hyperparameters:")
print(classification_report(y_test, y_pred_best_rf))
print(confusion_matrix(y_test, y_pred_best_rf))
```

```
Best hyperparameters: {'bootstrap': False, 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_e
Random Forest Classifier Results with Best Hyperparameters:
              precision    recall  f1-score   support

         0.0       0.96      0.98      0.97      9608
         1.0       0.92      0.83      0.88      2392

    accuracy                           0.95     12000
```

```
   macro avg       0.94      0.91      0.92     12000
weighted avg       0.95      0.95      0.95     12000

[[9445  163]
 [ 399 1993]]
```

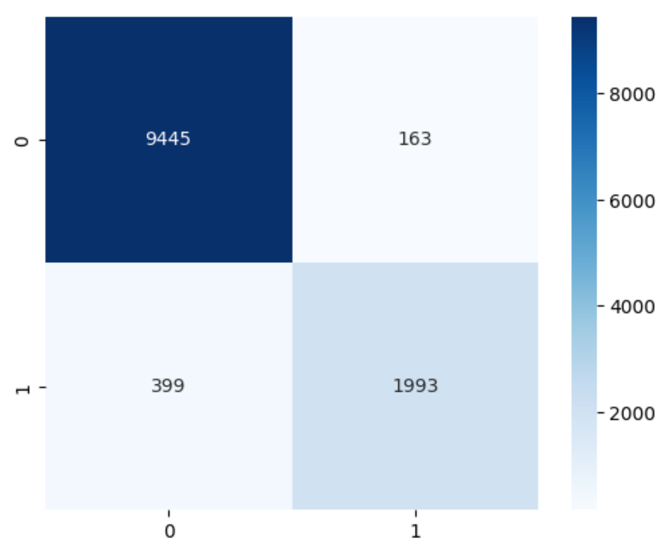
```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, y_pred_best_rf), annot=True, fmt='g', ax=ax);  #annot=True to annotate cells, ftm='g' to disable scienti
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['No Fraud', 'Yes Fraud']); ax.yaxis.set_ticklabels(['No Fraud', 'Yes Fraud']);
```




```
from sklearn.metrics import confusion_matrix
```

```
sns.heatmap(confusion_matrix(y_test, y_pred_best_rf), square=True, annot=True, cmap='Blues', fmt='d', cbar=True)
#agregar labels and tittle
```

<Axes: >



Interpretation:

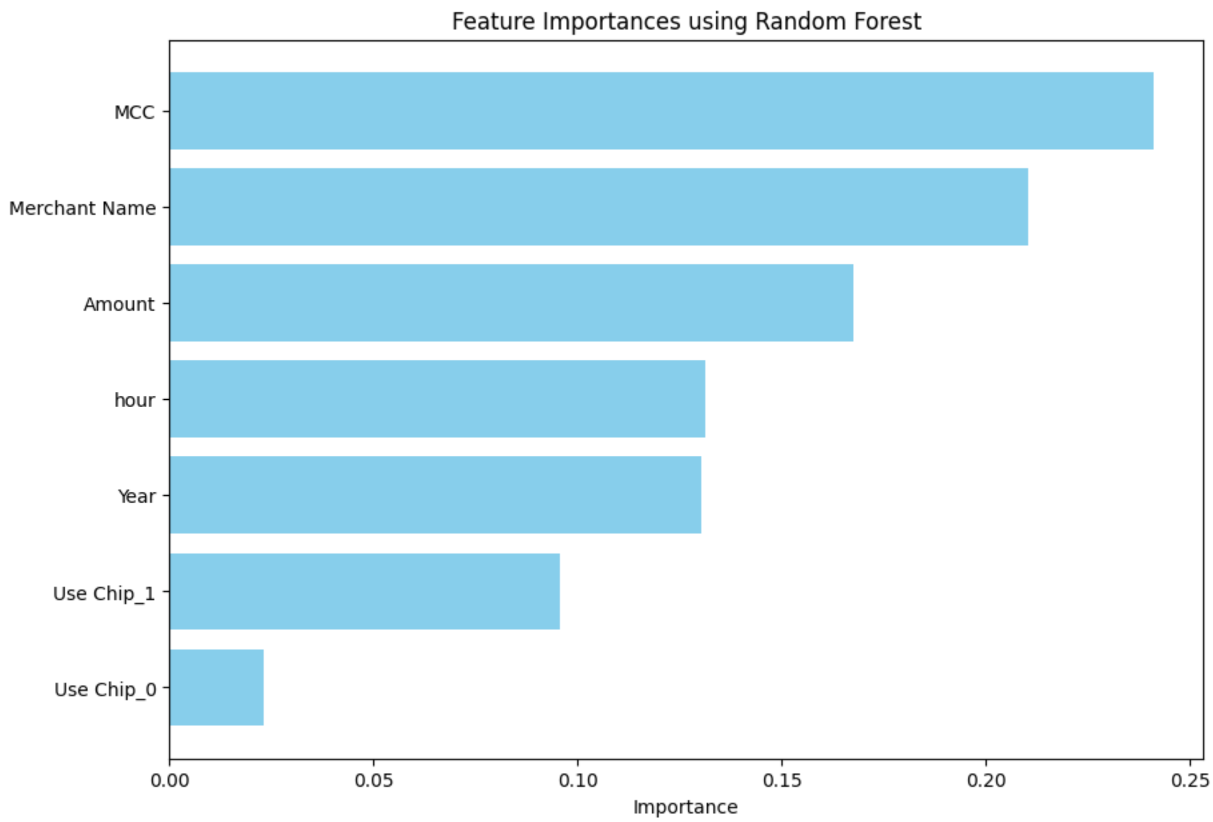
Precision: A precision of 0.92 for class 1 means that 92% of the predicted fraud cases were actually fraudulent.

Recall: A recall of 0.83 for class 1 means that the model identified 83% of the actual fraudulent transactions.

F1-score: A F1-score of 0.88 indicates a good balance between precision and recall.

In summary, the model achieved high accuracy (95%) and performed well in classifying non-fraudulent transactions (class 0). However, it showed relatively higher recall (98%) for fraudulent transactions (class 1), indicating that it missed some fraudulent cases. Overall, the model demonstrates a good performance but could be further improved to better detect fraud cases.

```
# Extract feature importances from the best random forest model
feature_importance = best_rf.feature_importances_
features = X_train.columns

# Sort the feature importances and their corresponding feature names
sorted_idx = feature_importance.argsort()

# Plot horizontal bar chart
plt.figure(figsize=(10, 7))
plt.barh(features[sorted_idx], feature_importance[sorted_idx], align='center', color='skyblue')
plt.xlabel('Importance')
plt.title('Feature Importances using Random Forest')
plt.show()
```



Insights

Based on the feature importance scores, the three top most important features for predicting fraud transactions are:

1. MCC: Merchant Category Code And Merchant Name • Insight: Certain types of merchants or industries may be more susceptible to fraudulent activities than others. Some businesses are at a higher risk of fraud than others. This can happen because they handle a lot of transactions, sell expensive items, or offer products that are popular in the illegal market. It's important to know which businesses might be more vulnerable to fraud
2. Amount: transaction amount • Insight: Fraudsters often use the size of a transaction to their advantage. They might make big purchases to get as much as they can before the card is reported stolen. Alternatively, they could make small purchases to check if the card works without attracting attention. So, if you notice transactions that are much bigger or smaller than your usual spending, it's a warning sign.

Recommendations

1. For MCC and Merchant Name: • Businesses and credit card companies need to pay extra attention to certain types of stores that are more likely to face fraud. They should keep a close watch and put extra safety measures in place for these places. It's also a good idea to

educate people about being careful when shopping in these specific areas.

2. For Amount: • Create a smart system that spots really high or low transactions compared to what a user usually spends. If something unusual happens, let users know about it