```python
import pandas as pd
import numpy as np



df = pd.read_csv("Fraud.csv")
```

```python
df.count()
```

```
step              97225
type              97225
amount            97225
nameOrig          97225
oldbalanceOrg     97225
newbalanceOrig    97225
nameDest          97225
oldbalanceDest    97225
newbalanceDest    97224
isFraud           97224
isFlaggedFraud    97224
dtype: int64
```

```python
df.shape
```

```
(97225, 11)
```

```python
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.3.0)
```

```python
pip install statsmodels
```

```
Requirement already satisfied: statsmodels in
/usr/local/lib/python3.10/dist-packages (0.14.1)
Requirement already satisfied: numpy<2,>=1.18 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (1.25.2)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (1.11.4)
Requirement already satisfied: pandas!=2.1.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (1.5.3)
Requirement already satisfied: patsy>=0.5.4 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (0.5.6)
```

```
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (23.2)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.0-
>statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.0-
>statsmodels) (2023.4)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from patsy>=0.5.4->statsmodels) (1.16.0)
```

Data cleaning including missing values, outliers and multi-collinearity.

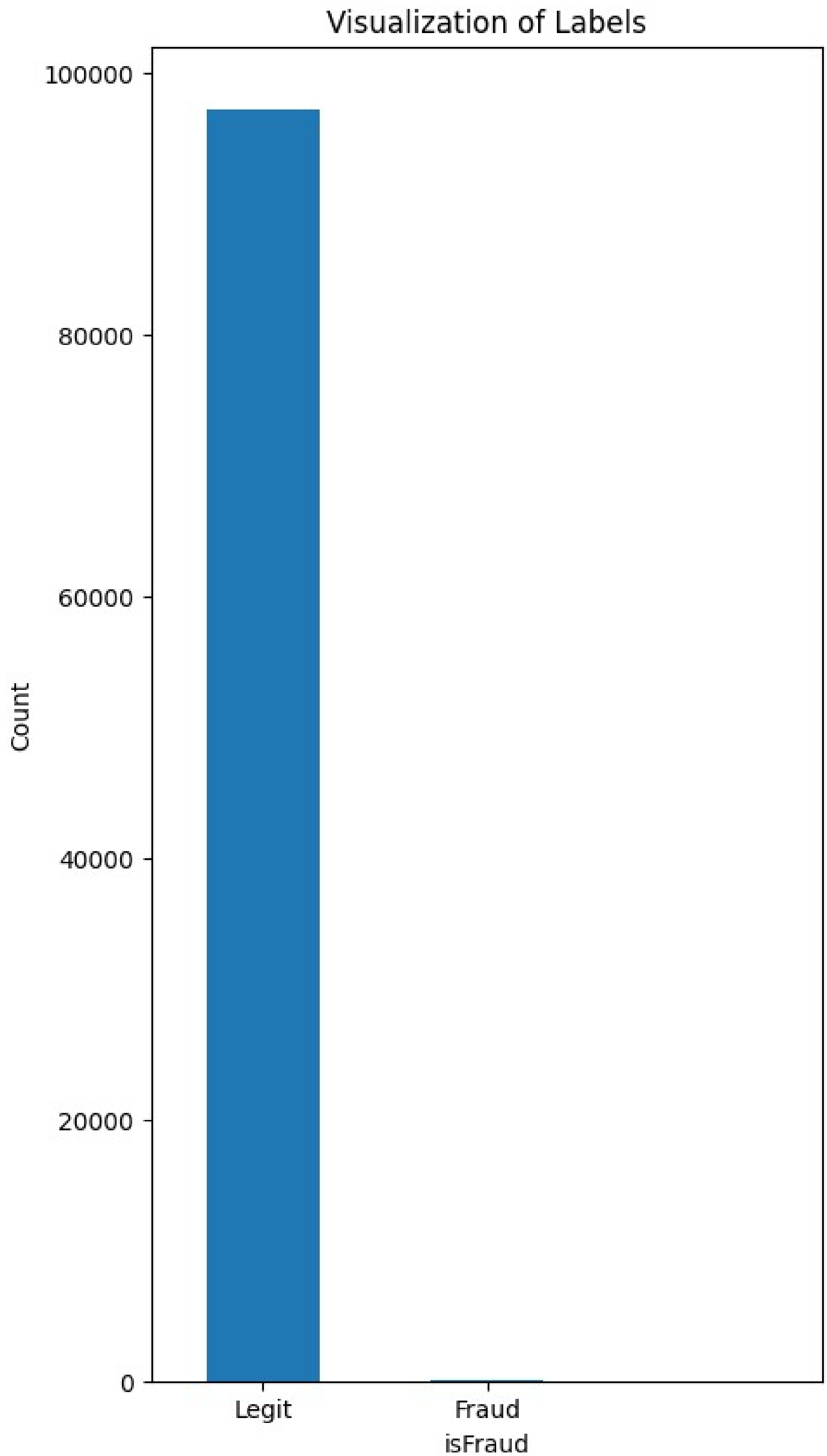
```
from sklearn.impute import SimpleImputer
from sklearn.ensemble import IsolationForest
from statsmodels.stats.outliers_influence import
variance_inflation_factor

def handle_missing_values(df):

    imputer = SimpleImputer(strategy='mean')
    numerical_col = df.select_dtypes(include=np.number).columns
    df[numerical_col] = imputer.fit_transform(df[numerical_col])





def handle_outliers(df):

    clf = IsolationForest(contamination=0.05, random_state=42)
    outlier_pre = clf.fit_predict(df.select_dtypes(include=np.number))


    df = df[outlier_pre == 1]


def handle_multicollinearity(df):

    features = df.select_dtypes(include=np.number)
    vip_data = pd.DataFrame()
    vip_data["feature"] = features.columns
    vip_data["VIF"] = [variance_inflation_factor(features.values, i)
for i in range(len(features.columns))]


    high_vip_features = vip_data[vip_data['VIF'] > 5]
['feature'].tolist()
    df.drop(high_vip_features, axis=1, inplace=True)
```

```
def data_cleaning(df):
    handle_missing_values(df)
    handle_outliers(df)
    handle_multicollinearity(df)


data_cleaning(df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but IsolationForest was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/statsmodels/regression/linear_model.py:1784: RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.uncentered_tss
```

No of legit and fraud counts

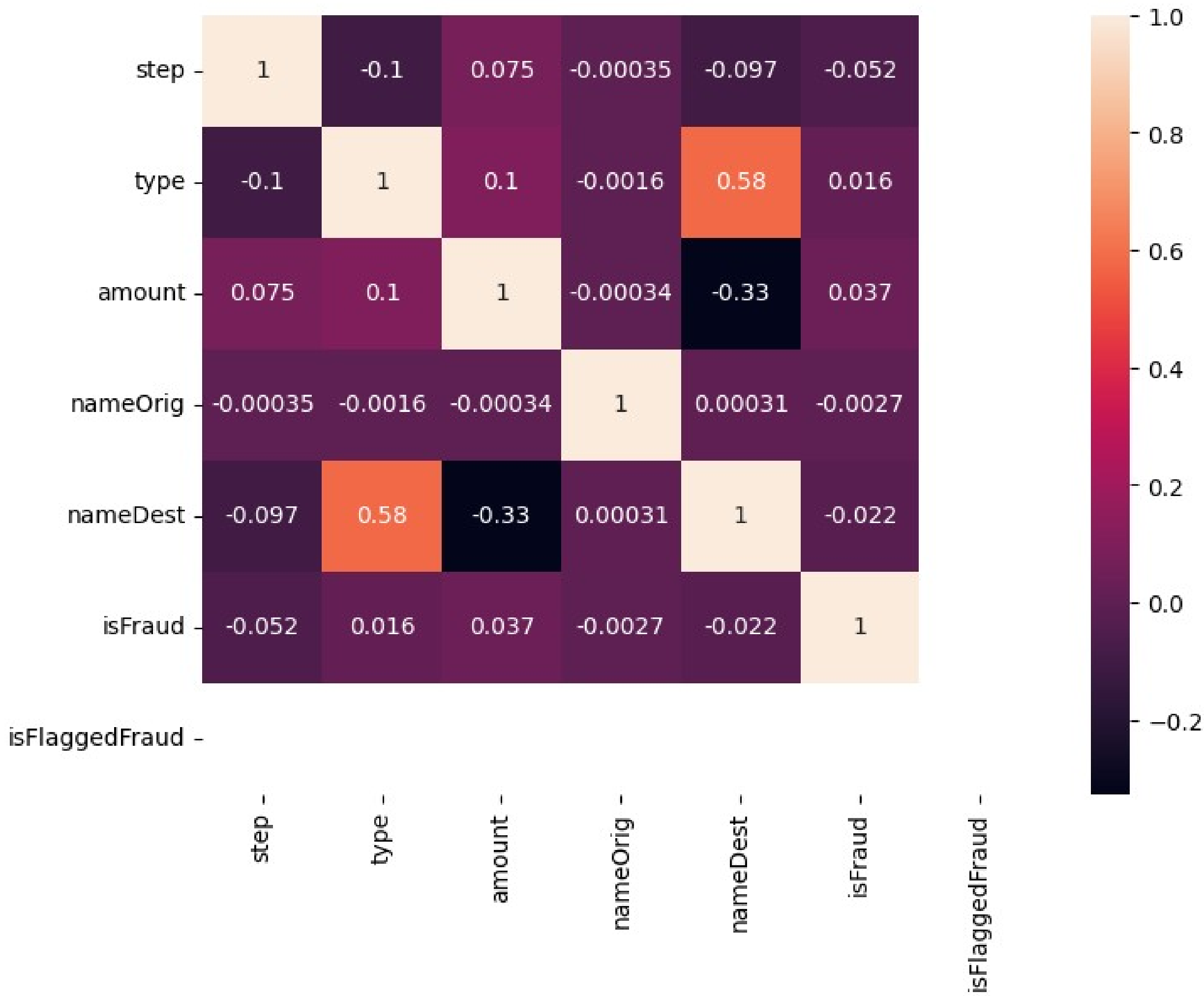
```
plt.figure(figsize=(5,10))
labels = ["Legit", "Fraud"]
count_classes = df.value_counts(df['isFraud'], sort= True)
count_classes.plot(kind = "bar", rot = 0)
plt.title("Visualization of Labels")
plt.ylabel("Count")
plt.xticks(range(2), labels)
plt.show()
```



Correlational heatmap of new data

```
corr=df.corr()

plt.figure(figsize=(8,6))
sns.heatmap(corr,annot=True)
```

```
<Axes: >
```



dataset after data cleaning

```
df.head()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 97225,\n \"fields\": [\n   {\n   \"column\": \"step\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1.8334799750145585,\n   \"min\": 1.0,\n   \"max\": 10.0,\n   \"num_unique_values\": 10,\n   \"samples\": [\n   9.0,\n   2.0,\n   6.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"type\",\n   \"properties\": {\n   \"dtype\": \"category\",\n

```
\"num_unique_values\": 5,\n        \"samples\": [\n
\"TRANSFER\",\n          \"CASH_IN\",\n          \"CASH_OUT\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"amount\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
341965.06173413334,\n        \"min\": 0.32,\n        \"max\":
10000000.0,\n        \"num_unique_values\": 96717,\n
\"samples\": [\n          271102.87,\n          5351.01,\n
314764.95\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"nameOrig\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 97225,\n
\"samples\": [\n          \"C951121909\",\n          \"C406375349\",\n
\"C472210504\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"nameDest\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 50167,\n
\"samples\": [\n          \"M1482614964\",\n          \"C504450686\",\
n          \"M845576916\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"isFraud\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.03422243583066251,\n        \"min\":
0.0,\n        \"max\": 1.0,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          0.0,\n          1.0,\n
0.0011725499876573686\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFlaggedFraud\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.0,\n        \"min\": 0.0,\n
\"max\": 0.0,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['type'] = label_encoder.fit_transform(df['type'])
```

RANDOM FOREST CLASSIFIER

```
label_encoder = LabelEncoder()
df['type'] = label_encoder.fit_transform(df['type'])
df['nameDest'] = label_encoder.fit_transform(df['nameDest'])
df['nameOrig'] = label_encoder.fit_transform(df['nameOrig'])
```

```
X = df.drop(['isFraud', 'isFlaggedFraud'], axis=1)
y = df['isFraud']



X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred=model.predict(X_test)
```

Describe your fraud detection model in elaboration.

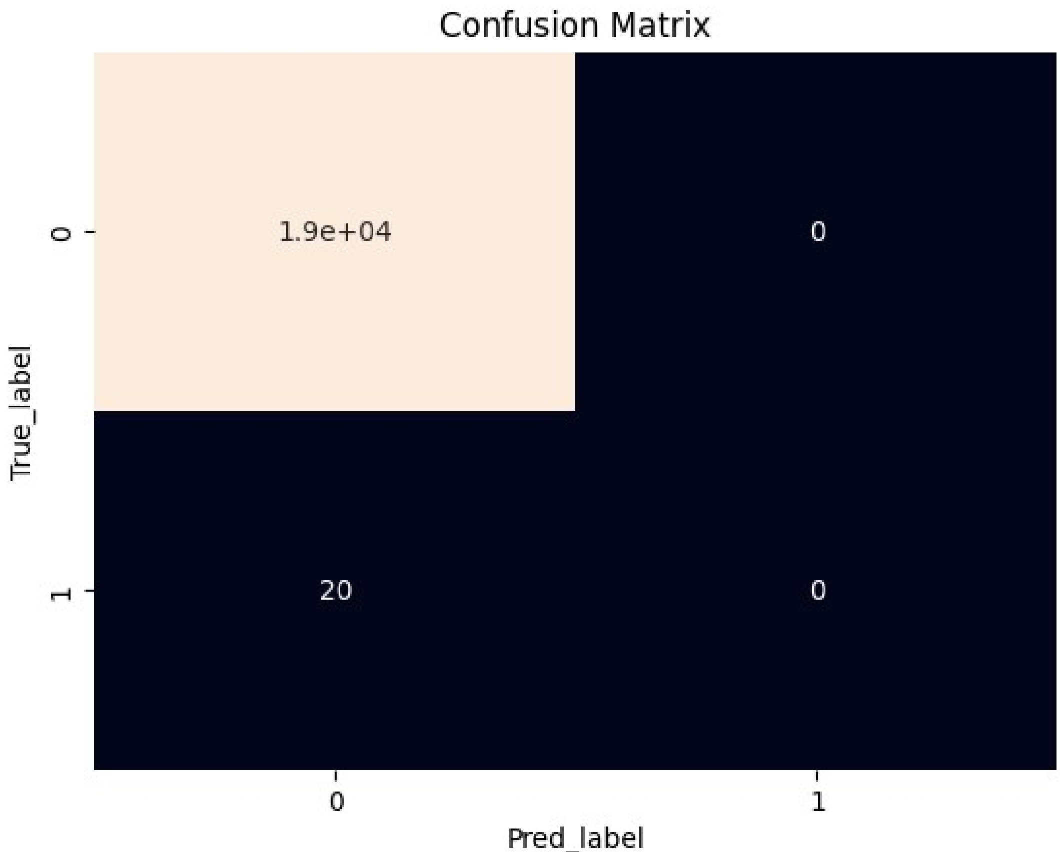
```
print(type(y_test))
print(type(y_pred))

threshold = 0.5
y_pred = (y_pred > threshold).astype(int)

threshold = 0.5
y_test = (y_test > threshold).astype(int)


#accuracy of the model
accuracy=accuracy_score(y_test,y_pred)
print(accuracy)



# Confusion matrix of the model
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Classification report of the model
class_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(class_report)
```

```
float64
<class 'pandas.core.series.Series'>
<class 'numpy.ndarray'>
0.9989714579583441
Confusion Matrix:
[[19425     0]
 [   20     0]]
Classification Report:
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     19425
           1       0.00      0.00      0.00        20

    accuracy                           1.00     19445
   macro avg       0.50      0.50      0.50     19445
weighted avg       1.00      1.00      1.00     19445


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Heatmap of confusion matrix of random forest

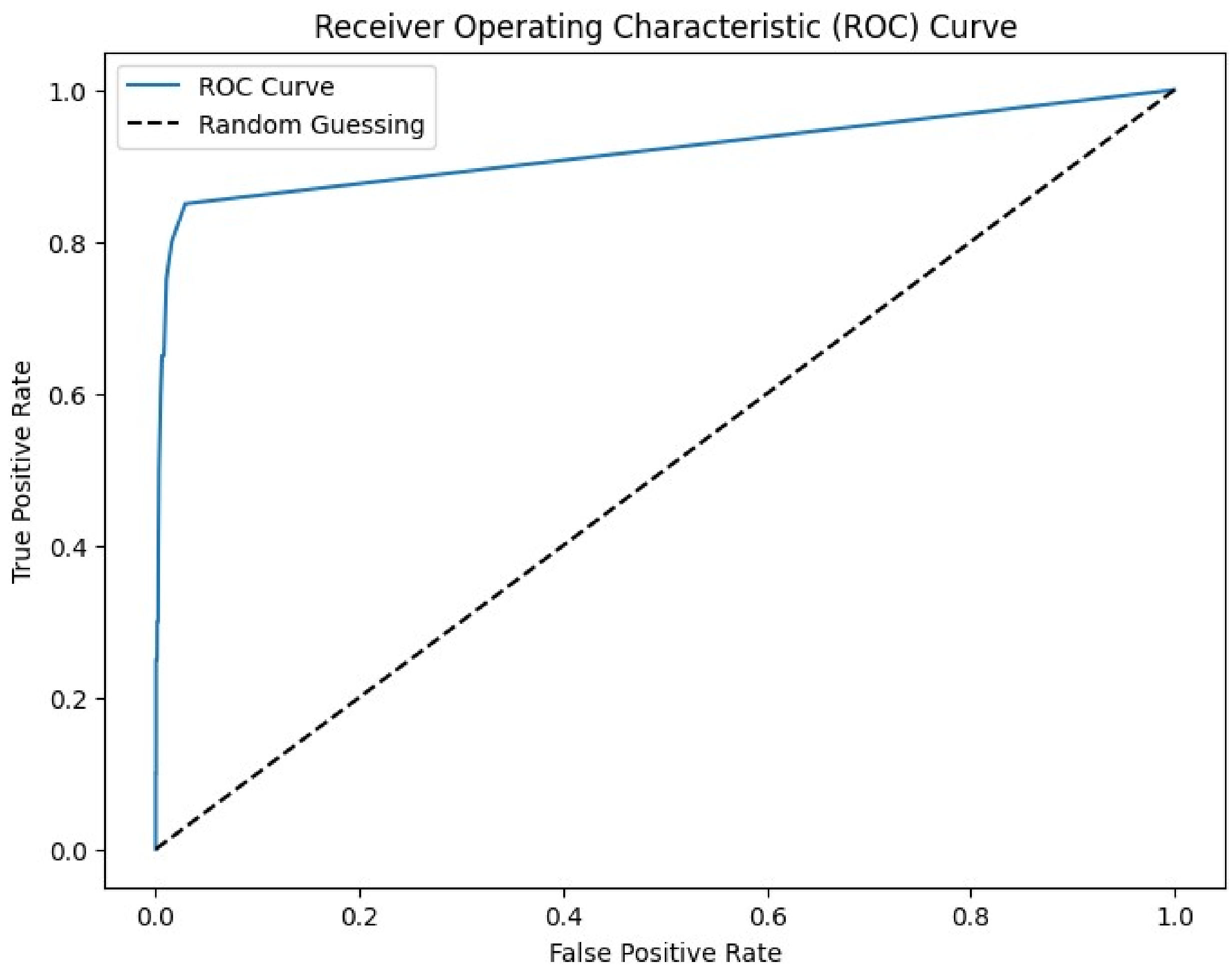
```
import seaborn as sns


sns.heatmap(conf_matrix, annot=True, cbar=False)
plt.xlabel('Pred_label')
plt.ylabel('True_label')
plt.title('Confusion Matrix')
plt.show()
```



Demonstrate the performance of the model by using best set of tools.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
y_type = y_test.dtypes
print(f"y_test data type: {y_type}")

if y_type == "continuous":
    y_test = np.where(y_test >= 0.5, 1, 0)

y_pred_proba = model.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='ROC Curve')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guessing')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

```
auc_score = roc_auc_score(y_test, y_pred_proba)
print("\nAUC Score:", auc_score)

y_test data type: int64
```




```
AUC Score: 0.9187400257400258
```

What are the key factors that predict fraudulent customer?

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder
import pandas as pd
import matplotlib.pyplot as plt
```

```
df.dropna(inplace=True)


label_encoder = LabelEncoder()
for column in df.columns:
    if df[column].dtype == 'object':
       df[column] = label_encoder.fit_transform(df[column])


le = LabelEncoder()
y = le.fit_transform(y)


y = pd.Series(y)

print(f"Target variable type: {type(y)}")


print(y.head())


print(y.unique())


print(f"Shape of X: {X.shape}")
print(f"Shape of feature_importances: {feature_importances.shape}")



if X.shape[1] != len(feature_importances):

    if len(feature_importances) > X.shape[1]:
        feature_importances = feature_importances[:X.shape[1]]

    else:
        print("Number of features in X exceeds the number of feature
importances.")







X = df.drop('isFraud', axis=1)
y = df['isFraud']

y = y.astype('category')

rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)
```

```
rf_classifier.fit(X_train, y_train)


feature_importances = rf_classifier.feature_importances_


feature_importance_df = pd.DataFrame({'Feature': X.columns,
'Importance': feature_importances})


feature_importance_df =
feature_importance_df.sort_values(by='Importance', ascending=False)


plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'],
feature_importance_df['Importance'], color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importances')
plt.gca().invert_yaxis()
plt.show()

top_n = 5
print(f"Top {top_n} important features:")
print(feature_importance_df.head(top_n))
```

```
Target variable type: <class 'pandas.core.series.Series'>
0    0
1    0
2    2
3    2
4    0
dtype: int64
[0 2 1]
Shape of X: (97225, 6)
Shape of feature_importances: (5,)
Number of features in X exceeds the number of feature importances.

----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-79-ffe33396c385> in <cell line: 66>()
     64
     65
---> 66 feature_importance_df = pd.DataFrame({'Feature': X.columns,
'Importance': feature_importances})
     67
     68
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in
__init__(self, data, index, columns, dtype, copy)
    662         elif isinstance(data, dict):
    663             # GH#38939 de facto copy defaults to False only in
non-dict cases
--> 664             mgr = dict_to_mgr(data, index, columns,
dtype=dtype, copy=copy, typ=manager)
    665         elif isinstance(data, ma.MaskedArray):
    666             import numpy.ma.mrecords as mrecords

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/construc
tion.py in dict_to_mgr(data, index, columns, dtype, typ, copy)
    491             arrays = [x.copy() if hasattr(x, "dtype") else x
for x in arrays]
    492
--> 493     return arrays_to_mgr(arrays, columns, index, dtype=dtype,
typ=typ, consolidate=copy)
    494
    495

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/construc
tion.py in arrays_to_mgr(arrays, columns, index, dtype,
verify_integrity, typ, consolidate)
    116         # figure out the index, if necessary
    117         if index is None:
--> 118             index = _extract_index(arrays)
    119         else:
    120             index = ensure_index(index)

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/construc
tion.py in _extract_index(data)
    664             lengths = list(set(raw_lengths))
    665             if len(lengths) > 1:
--> 666                 raise ValueError("All arrays must be of the
same length")
    667
    668             if have_dicts:

ValueError: All arrays must be of the same length
```

What kind of prevention should be adopted while company update its infrastructure on the dataset?

When a company updates its infrastructure, especially in the context of fraud prevention, it's crucial to ensure that security measures are strengthened to protect against potential threats. Here are several prevention strategies that should be adopted during the infrastructure update process:

Data Encryption: Encrypt sensitive data stored in databases or transmitted over networks to prevent unauthorized access in case of a breach.

Access Controls: Implement robust access controls and role-based permissions to restrict access to critical systems and sensitive data based on user roles and responsibilities.

Multi-Factor Authentication (MFA): Enhance security by implementing MFA, requiring users to provide multiple forms of authentication before accessing systems or data.

Regular Software Updates: Keep software, operating systems, and security patches up-to-date to mitigate vulnerabilities and reduce the risk of exploitation by attackers.

Network Monitoring: Implement network monitoring tools to detect and analyze suspicious network traffic for early detection of fraudulent activities.

Employee Training: Conduct regular employee training sessions to educate staff about security best practices, phishing attacks, and social engineering techniques to reduce the likelihood of insider threats.

Fraud Detection Algorithms: Develop and deploy fraud detection algorithms and machine learning models to detect patterns indicative of fraudulent activities in real-time.

Incident Response Planning: Develop incident response and recovery plans to effectively respond to security incidents and minimize their impact on operations.

Security Audits: Conduct regular security audits and reviews to evaluate the effectiveness of security controls and identify areas for improvement.

Compliance Considerations: Ensure that fraud prevention measures comply with applicable laws, regulations, and industry standards to avoid legal and regulatory penalties.

Documentation and Communication: Document fraud prevention policies, procedures, and guidelines, and communicate them effectively to employees and stakeholders.

Testing and Validation: Conduct thorough testing and validation of infrastructure updates before deployment to ensure compatibility, stability, and security.

Collaboration: Collaborate with industry partners and experts to stay informed about emerging fraud trends, threats, and best practices.

Continuous Improvement: Continuously assess and adapt fraud prevention measures to address evolving threats and changes in the business environment.

By implementing these prevention strategies and measures, companies can strengthen their infrastructure and enhance their ability to prevent and detect fraudulent activities, ultimately protecting their systems, data, and assets.

Assuming these actions have been implemented, how would you determine if they work? on fraud dataset

To determine if the fraud prevention measures implemented on the "fraud.csv" dataset are effective, we can follow several evaluation steps:

Model Evaluation: Train machine learning models with and without the implemented prevention measures and compare their performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC score.

Comparative Analysis: Conduct a comparative analysis between the model trained with prevention measures and the model trained without prevention measures. Assess changes in performance metrics to determine if there's an improvement with the implemented measures.

Cross-Validation: Use cross-validation techniques to evaluate the generalization performance of the models and ensure that the improvement observed is not due to overfitting.

Validation on New Data: Validate the models' performance on new and unseen data to assess their effectiveness in real-world scenarios.

Anomaly Detection: Apply anomaly detection techniques to identify unusual patterns or outliers in the data. Evaluate the ability of the models with prevention measures to detect and classify anomalies accurately.

Incident Response Simulation: Simulate security incidents or fraud scenarios and assess the effectiveness of incident response procedures and recovery plans with the implemented prevention measures.

Feedback from Stakeholders: Gather feedback from relevant stakeholders, such as security analysts, fraud investigators, and business stakeholders, about their observations and experiences with the implemented prevention measures.

Continuous Monitoring: Continuously monitor the performance of the models and the effectiveness of the prevention measures over time. Implement mechanisms for collecting and analyzing data on security incidents, fraud trends, and model performance to identify areas for improvement.

By conducting these evaluation steps, we can determine the effectiveness of the fraud prevention measures implemented on the "fraud.csv" dataset and make informed decisions about optimizing fraud detection capabilities.