```
#import the libraries
from google.colab import drive
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_selection import mutual_info_classif
from sklearn.feature_selection import SelectKBest
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, precision_recall_curve
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#read the dataset
drive.mount('/content/drive')
df_train = pd.read_csv("/content/drive/MyDrive/archive/fraudTrain.csv")
df_test = pd.read_csv("/content/drive/MyDrive/archive/fraudTest.csv")
```

```
Mounted at /content/drive
```

```
#read the dataset
df = pd.concat([df_train, df_test], ignore_index=True, axis=0)
df.head()
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	...	36.0
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	48.8
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	42.1
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	...	46.2
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	...	38.4

5 rows × 23 columns

```
df.describe()
```

	Unnamed: 0	cc_num	amt	zip	lat	l
count	1.852394e+06	1.852394e+06	1.852394e+06	1.852394e+06	1.852394e+06	1.852394e
mean	5.371934e+05	4.173860e+17	7.006357e+01	4.881326e+04	3.853931e+01	-9.022783e
std	3.669110e+05	1.309115e+18	1.592540e+02	2.688185e+04	5.071470e+00	1.374789e
min	0.000000e+00	6.041621e+10	1.000000e+00	1.257000e+03	2.002710e+01	-1.656723e
25%	2.315490e+05	1.800429e+14	9.640000e+00	2.623700e+04	3.466890e+01	-9.679800e
50%	4.630980e+05	3.521417e+15	4.745000e+01	4.817400e+04	3.935430e+01	-8.747690e
75%	8.335758e+05	4.642255e+15	8.310000e+01	7.204200e+04	4.194040e+01	-8.015800e
max	1.296674e+06	4.992346e+18	2.894890e+04	9.992100e+04	6.669330e+01	-6.795030e

```
#legit and fraudlent
print(df['is_fraud'].value_counts())
legit = df[df.is_fraud==0]
fraudlent = df[df.is_fraud==1]
print(f"legit data: {legit.shape}")
print(f"fraudlent data: {fraudlent.shape}")
```

```
0    1842743
1       9651
Name: is_fraud, dtype: int64
legit data: (1842743, 23)
fraudlent data: (9651, 23)
```

```
#under sampling
legit = legit.sample(n=9700)
print(f"legit data: {legit.shape}")
print(f"fraudlent data: {fraudlent.shape}")
df=pd.concat([legit,fraudlent],axis=0)
print(df.sort_index())
```

```
legit data: (9700, 23)
fraudlent data: (9651, 23)
         Unnamed: 0 trans_date_trans_time              cc_num  \
176             176   2019-01-01 02:10:12     4740713119940984
271             271   2019-01-01 03:30:49       36078114201167
360             360   2019-01-01 04:42:23       38588538868506
506             506   2019-01-01 06:39:59     3590736522064285
659             659   2019-01-01 08:36:25     3583635130604947
...             ...                   ...                  ...
1851288      554613   2020-12-31 17:37:10     4265776278887457
1851454      554779   2020-12-31 18:28:41  4223708906367574214
1851793      555118   2020-12-31 20:30:50      213161231269724
1851883      555208   2020-12-31 21:02:06       30030380240193
1852105      555430   2020-12-31 22:15:27       30407675418785

                                   merchant        category     amt  \
176                    fraud_Brown-Greenholt   entertainment    5.27
271                    fraud_Deckow-O'Conner     grocery_pos  145.21
360      fraud_Schaefer, Maggio and Daugherty   gas_transport   60.20
506                    fraud_Stamm-Rodriguez        misc_pos   76.40
659                       fraud_Bauch-Raynor     grocery_pos  206.85
...                                      ...             ...     ...
1851288                   fraud_Harris Group     food_dining   91.13
1851454         fraud_Hauck, Dietrich and Funk       kids_pets   49.45
1851793                    fraud_Gerhold LLC            home   32.20
1851883      fraud_Hyatt, Russel and Gleichner  health_fitness   76.44
1852105                   fraud_Kris-Padberg    shopping_pos   12.88

               first      last gender                     street  ...  \
176          Heather     Hines      F           13776 Hicks Plains  ...
271      Christopher      Horn      M          956 Sanchez Highway  ...
360       Jacqueline     Curry      F              3047 Jeff Place  ...
506         Kimberly  Gonzalez      F  72966 Shannon Pass Apt. 391  ...
659          Crystal    Gamble      F   899 Michele View Suite 960  ...
...              ...       ...    ...                         ...  ...
1851288    Christine      Best      F            68248 Deanna Land  ...
1851454         Adam    Riddle      M            27718 Mason Bypass  ...
1851793       Alyssa    Morgan      F       622 Robin Run Suite 764  ...
1851883      William   Jenkins      M            50614 Kevin Point  ...
1852105     Danielle     Evans      F   76752 David Lodge Apt. 064  ...

             lat      long  city_pop                      job         dob  \
176      41.1901  -74.0436      9993      Pensions consultant  1962-10-16
271      37.2692  -82.9161       798      Facilities manager  1926-06-26
360      30.1886 -103.2214       498            Lexicographer  1990-11-23
506      34.5091  -92.4828      4074  Scientist, audiological  1975-12-20
659      40.0369  -75.0664   1526206      Structural engineer  1985-01-01
...          ...       ...       ...                      ...         ...
1851288  35.2087  -92.2123       969       Physicist, medical  1954-01-05
1851454  39.0965  -84.6431       177      Exhibition designer  1974-05-30
1851793  34.0480  -85.9246     67082  Physiological scientist  1963-02-09
1851883  30.2816  -99.2410      2395    Pharmacist, community  1993-11-17
1852105  42.1939  -76.7361       520          Psychotherapist  1991-10-13

                                trans_num   unix_time  merch_lat  merch_long  \
176      9f55bd65f64193f1b1a46a99c93e7844  1325383812  41.788775  -74.471154
271      50fd631627ff1c488113e8c9249ab801  1325388649  37.693205  -82.671226
360      9da7e99fc8147da68b5e322d7c63c096  1325392943  29.335557 -103.852785
```

```
#statistical measures
print(df.describe())
print(df.groupby('is_fraud').mean())
```

```
         Unnamed: 0        cc_num           amt           zip           lat  \
count  1.935100e+04  1.935100e+04  19351.000000  19351.000000  19351.000000
mean   5.386690e+05  3.975616e+17    298.099432  48464.971423     38.663691
std    3.792391e+05  1.277992e+18    375.506985  27079.031702      5.111601
min    1.760000e+02  6.041621e+10      1.000000   1257.000000     20.027100
25%    2.150450e+05  1.800365e+14     20.170000  25213.000000     34.847000
50%    4.593270e+05  3.520550e+15     88.850000  47863.000000     39.433600
75%    8.667740e+05  4.633065e+15    465.165000  71762.000000     42.074000
max    1.296623e+06  4.992346e+18   8517.380000  99921.000000     66.693300

               long      city_pop     unix_time     merch_lat    merch_long  \
```

```
count  19351.000000  1.935100e+04  1.935100e+04  19351.000000  19351.000000
mean     -90.209527  8.794817e+04  1.357116e+09     38.661089    -90.210209
std       14.130837  2.979177e+05  1.817924e+07      5.145979     14.144194
min     -165.672300  2.300000e+01  1.325384e+09     19.121455   -166.562839
25%      -96.786900  7.540000e+02  1.341615e+09     34.930664    -96.841426
50%      -87.349000  2.526000e+03  1.356332e+09     39.473030    -87.281755
75%      -80.065200  1.940800e+04  1.372604e+09     42.016427    -80.102078
max      -67.950300  2.906700e+06  1.388528e+09     67.510267    -66.960745

           is_fraud
count  19351.000000
mean       0.498734
std        0.500011
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        1.000000
            Unnamed: 0        cc_num         amt           zip        lat  \
is_fraud
0        537494.653918  3.980812e+17   66.712249  48927.953814  38.584969
1        539849.247228  3.970393e+17  530.661412  47999.638379  38.742813

               long      city_pop     unix_time  merch_lat  merch_long
is_fraud
0        -90.384436  85908.282784  1.358900e+09  38.587590  -90.381628
1        -90.033730  89998.422961  1.355323e+09  38.734962  -90.037919
<ipython-input-7-c5a531823b49>:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futu
  print(df.groupby('is_fraud').mean())
```

```
#print number of unique categories for every attribute
print(f"Total number of rows: {df.shape[0]}")
for col in df.columns:
  print(f'{col}: {df[col].nunique()}')
```

```
Total number of rows: 19351
Unnamed: 0: 19300
trans_date_trans_time: 19345
cc_num: 999
merchant: 693
category: 14
amt: 13994
first: 355
last: 486
gender: 2
street: 999
city: 906
state: 51
zip: 985
lat: 983
long: 983
city_pop: 891
job: 497
dob: 984
trans_num: 19351
unix_time: 19345
merch_lat: 19338
merch_long: 19345
is_fraud: 2
```

```
#dropping columns and check for null values
df.drop(labels=['Unnamed: 0'], axis=1, inplace=True)
print(f"Null values: {df.isnull().values.any()}")
print(df.columns)
```

```
Null values: False
Index(['trans_date_trans_time', 'cc_num', 'merchant', 'category', 'amt',
       'first', 'last', 'gender', 'street', 'city', 'state', 'zip', 'lat',
       'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time', 'merch_lat',
       'merch_long', 'is_fraud'],
      dtype='object')
```

```
#calculate entropy for taget variable
target = "is_fraud"
probabilities = df[target].value_counts(normalize=True)/len(df)
entropy = -np.sum(probabilities*np.log2(probabilities))
print(f"{target}: {entropy}")
```

```
is_fraud: 0.0007875621867953986
```

```
#change string labels to numerical
label_encoder = LabelEncoder()
stringTypeColumns = ['trans_date_trans_time','cc_num', 'merchant', 'category', 'first', 'last', 'gender', 'street', 'city', 'state', 'z:
for column in stringTypeColumns:
    df[column] = label_encoder.fit_transform(df[column])
```

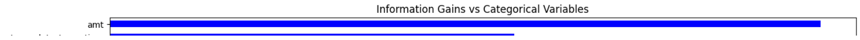
```
#calculate information gain for the features
X=df.drop('is_fraud', axis=1)
y=df['is_fraud']
information_gain = mutual_info_classif(X, y, discrete_features='auto', random_state=1)
information_gain
```

```
array([0.25748812, 0.08298698, 0.09281903, 0.08651023, 0.4525601 ,
       0.03049561, 0.03859248, 0.00135004, 0.07616792, 0.07114658,
       0.00488369, 0.07923783, 0.07601854, 0.07323542, 0.06354338,
       0.03173214, 0.0775402 , 0.00234236, 0.24890171, 0.00185617,
       0.0011095 ])
```

```
#print the information gain
info_gain = pd.Series(information_gain)
info_gain.index = X.columns
info_gain.sort_values(ascending= False)
```

```
amt                      0.452560
trans_date_trans_time    0.257488
unix_time                0.248902
merchant                 0.092819
category                 0.086510
cc_num                   0.082987
zip                      0.079238
dob                      0.077540
street                   0.076168
lat                      0.076019
long                     0.073235
city                     0.071147
city_pop                 0.063543
last                     0.038592
job                      0.031732
first                    0.030496
state                    0.004884
trans_num                0.002342
merch_lat                0.001856
gender                   0.001350
merch_long               0.001109
dtype: float64
```

```
#plot the information gain graph
ax = info_gain.sort_values(ascending = True).plot(kind='barh', color='blue', figsize=(16, 6))

plt.xlabel('Information Gains')
plt.ylabel('Categorical Variables')
plt.title('Information Gains vs Categorical Variables')

plt.show()
```




```
#selecting features and create new dataframe
select_cols = SelectKBest(mutual_info_classif, k=10)
select_cols.fit_transform(X,y)
selected_indices = select_cols.get_support(indices=True)
selected_indices = list(selected_indices) + [df.columns.get_loc('is_fraud')]
new_df = df.iloc[:, selected_indices]
print(new_df.sort_index())
```

```
              trans_date_trans_time  cc_num  merchant  category     amt  street  \
176                               0     767        81         0    5.27     134
271                               1     219       130         4  145.21     963
360                               2     238       548         2   60.20     314
506                               3     627       592         9   76.40     738
659                               4     616        29         4  206.85     908
...                             ...     ...       ...       ...     ...     ...
1851288                       19340     688       230         1   91.13     692
1851454                       19341     924       234         7   49.45     279
1851793                       19342     295       188         6   32.20     636
1851883                       19343     157       274         5   76.44     517
1852105                       19344     198       341        12   12.88     785

         zip      lat      long   unix_time  is_fraud
176       70  41.1901  -74.0436  1325383812         0
271      427  37.2692  -82.9161  1325388649         0
360      844  30.1886 -103.2214  1325392943         0
506      745  34.5091  -92.4828  1325399999         0
659      182  40.0369  -75.0664  1325406985         0
...      ...      ...       ...         ...       ...
1851288  747  35.2087  -92.2123  1388511430         0
1851454  458  39.0965  -84.6431  1388514521         0
1851793  360  34.0480  -85.9246  1388521850         0
1851883  831  30.2816  -99.2410  1388523726         0
1852105  130  42.1939  -76.7361  1388528127         0

[19351 rows x 11 columns]
```

```
#print new dataframe statistical measures
print(f"Total number of rows: {new_df.shape[0]}")
for col in new_df.columns:
  print(f'{col}: {new_df[col].nunique()}')
print(new_df.describe())
print(new_df.groupby('is_fraud').mean())
```

```
Total number of rows: 19351
trans_date_trans_time: 19345
cc_num: 999
merchant: 693
category: 14
amt: 13994
street: 999
zip: 985
lat: 983
long: 983
unix_time: 19345
is_fraud: 2
       trans_date_trans_time        cc_num      merchant      category  \
count           19351.000000  19351.000000  19351.000000  19351.000000
mean             9672.016537    497.293266    341.036949      6.734277
std              5584.208423    286.417764    197.287791      3.870311
min                 0.000000      0.000000      0.000000      0.000000
25%              4836.500000    249.000000    173.000000      4.000000
50%              9672.000000    493.000000    345.000000      7.000000
75%             14506.500000    744.000000    504.000000     11.000000
max             19344.000000    998.000000    692.000000     13.000000

                amt        street           zip           lat          long  \
count  19351.000000  19351.000000  19351.000000  19351.000000  19351.000000
mean     298.099432    497.363030    491.162421     38.663691    -90.209527
std      375.506985    286.954735    285.970290      5.111601     14.130837
min        1.000000      0.000000      0.000000     20.027100   -165.672300
25%       20.170000    253.000000    242.000000     34.847000    -96.786900
50%       88.850000    495.000000    493.000000     39.433600    -87.349000
75%      465.165000    743.000000    741.000000     42.074000    -80.065200
max     8517.380000    998.000000    984.000000     66.693300    -67.950300

          unix_time      is_fraud
count  1.935100e+04  19351.000000
mean   1.357116e+09      0.498734
std    1.817924e+07      0.500011
min    1.325384e+09      0.000000
25%    1.341615e+09      0.000000
50%    1.356332e+09      0.000000
75%    1.372604e+09      1.000000
max    1.388528e+09      1.000000
          trans_date_trans_time      cc_num    merchant  category         amt  \
```

```
is_fraud
0                 10217.469278  500.465464  340.328969  6.198247   66.712249
1                  9123.794425  494.104963  341.748523  7.273029  530.661412

              street         zip        lat       long     unix_time
is_fraud
0         497.153299  496.396598  38.584969 -90.384436  1.358900e+09
1         497.573827  485.901668  38.742813 -90.033730  1.355323e+09
```

```
# perform Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(new_df.drop('is_fraud', axis=1), new_df['is_fraud'], test_size=0.3, random_state=41)
```

```
# Perform Grid Search Cross-Validation
dt_model = DecisionTreeClassifier()
param_grid = {
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [5, 10, 15],
    'min_samples_leaf': [2, 4, 8]
}
grid_search = GridSearchCV(estimator=dt_model, param_grid=param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print(best_params)
```

```
{'max_depth': 10, 'min_samples_leaf': 8, 'min_samples_split': 5}
```

```
# Create Decision Tree model with the best hyperparameters
best_dt_model = DecisionTreeClassifier(**best_params)
best_dt_model.fit(X_train, y_train)
y_pred = best_dt_model.predict(X_test)
```

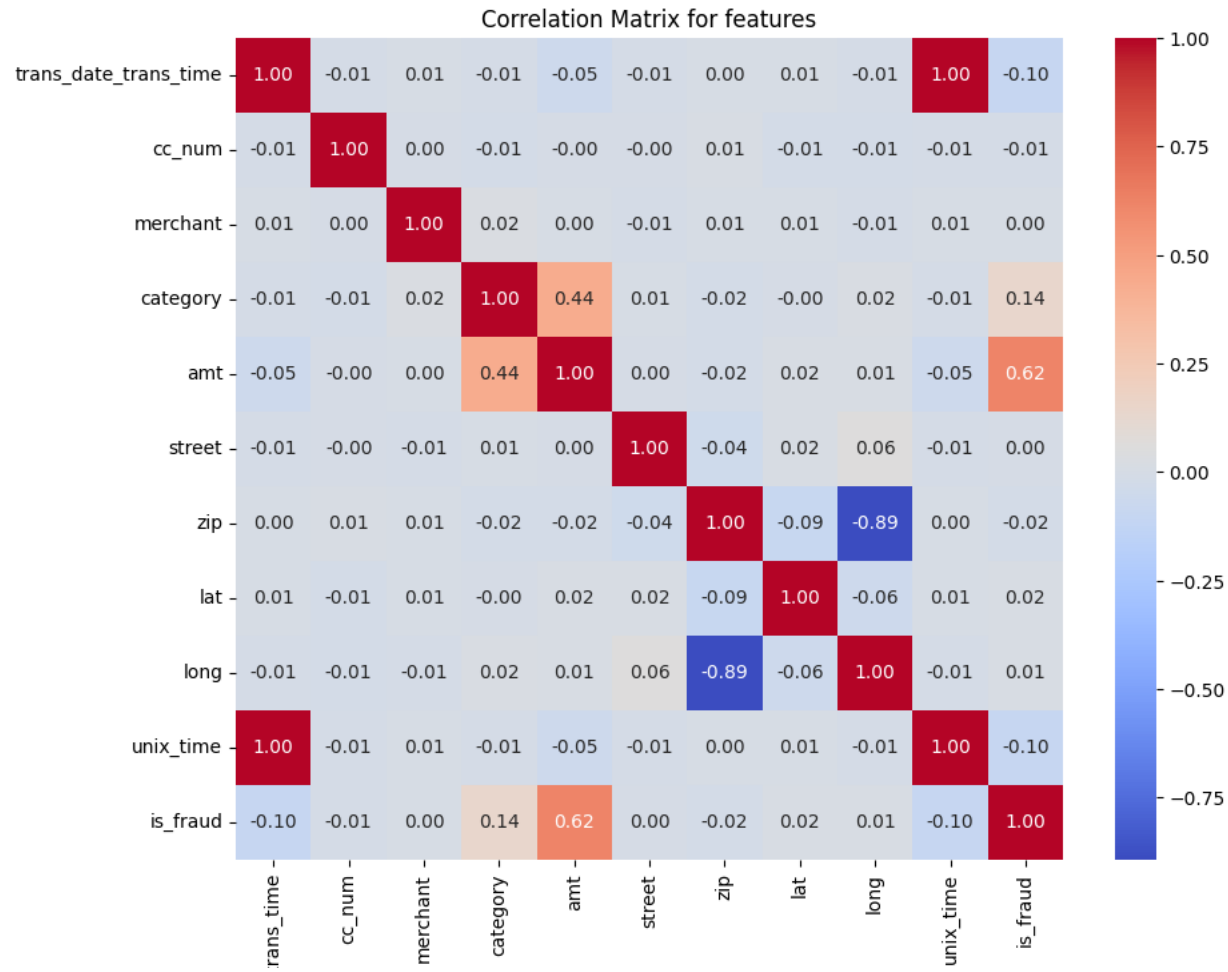
```
# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.961591457113331
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.95      0.96      2893
           1       0.96      0.97      0.96      2913

    accuracy                           0.96      5806
   macro avg       0.96      0.96      0.96      5806
weighted avg       0.96      0.96      0.96      5806

Confusion Matrix:
 [[2761  132]
 [  91 2822]]
```

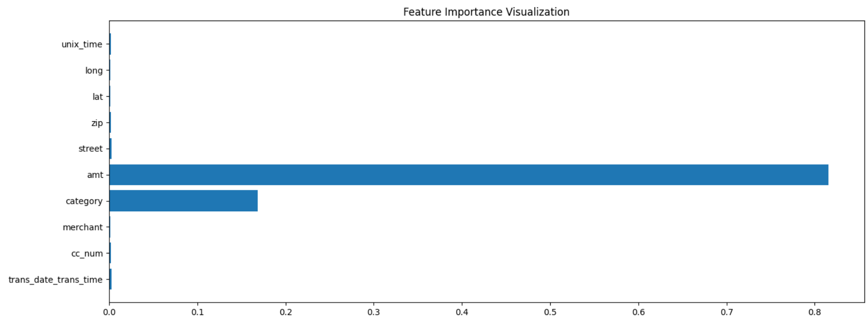
```
# Display feature importances
feature_importances = best_dt_model.feature_importances_
print("Feature Importances:")
for feature, importance in zip(X_train.columns, feature_importances):
    print(f"{feature}: {importance:.5f}")
```

```
Feature Importances:
trans_date_trans_time: 0.00292
cc_num: 0.00198
merchant: 0.00123
category: 0.16852
amt: 0.81564
street: 0.00274
zip: 0.00216
lat: 0.00137
long: 0.00166
unix_time: 0.00177
```

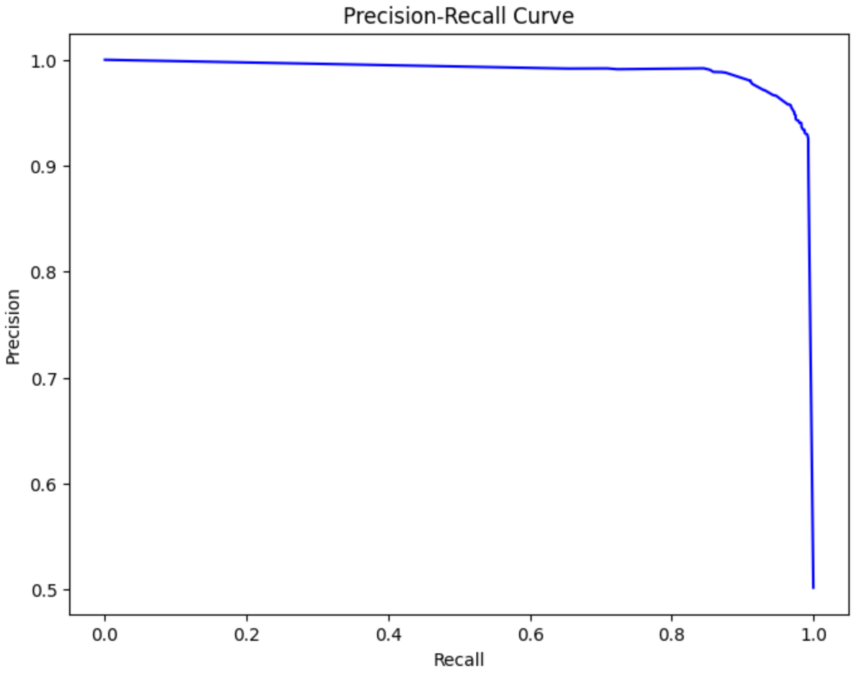
```
#plot correlaion matrix
correlation_matrix=new_df.corr()
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Matrix for features")
plt.show()
```



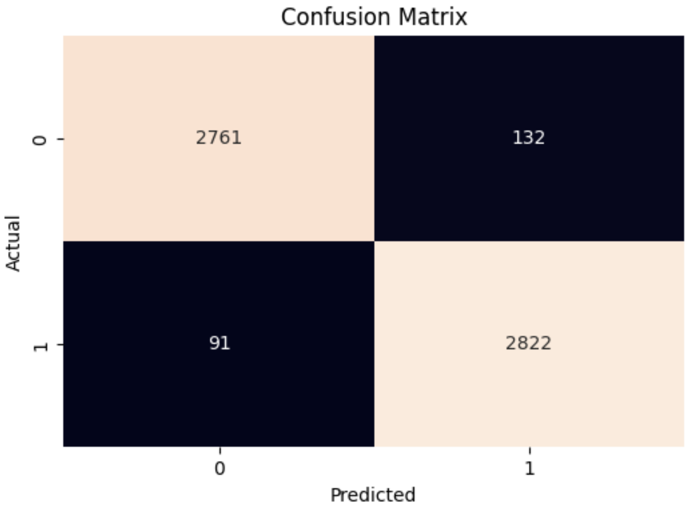

```
#plot feature importances
plt.figure(figsize=(16, 6))
plt.barh(X_train.columns, best_dt_model.feature_importances_)
plt.title('Feature Importance Visualization')
plt.show()
```



```
# Plot the precision-recall curve
precision, recall, _ = precision_recall_curve(y_test, best_dt_model.predict_proba(X_test)[:, 1])
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```




```
# Plot the confusion matrix using heatmap
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='g', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```




```
# Visualize the decision tree
plt.figure(figsize=(150, 100))
plot_tree(best_dt_model, filled=True, feature_names=[f"Feature {i}" for i in range(X.shape[1])], class_names=["0", "1"])
plt.show()
```