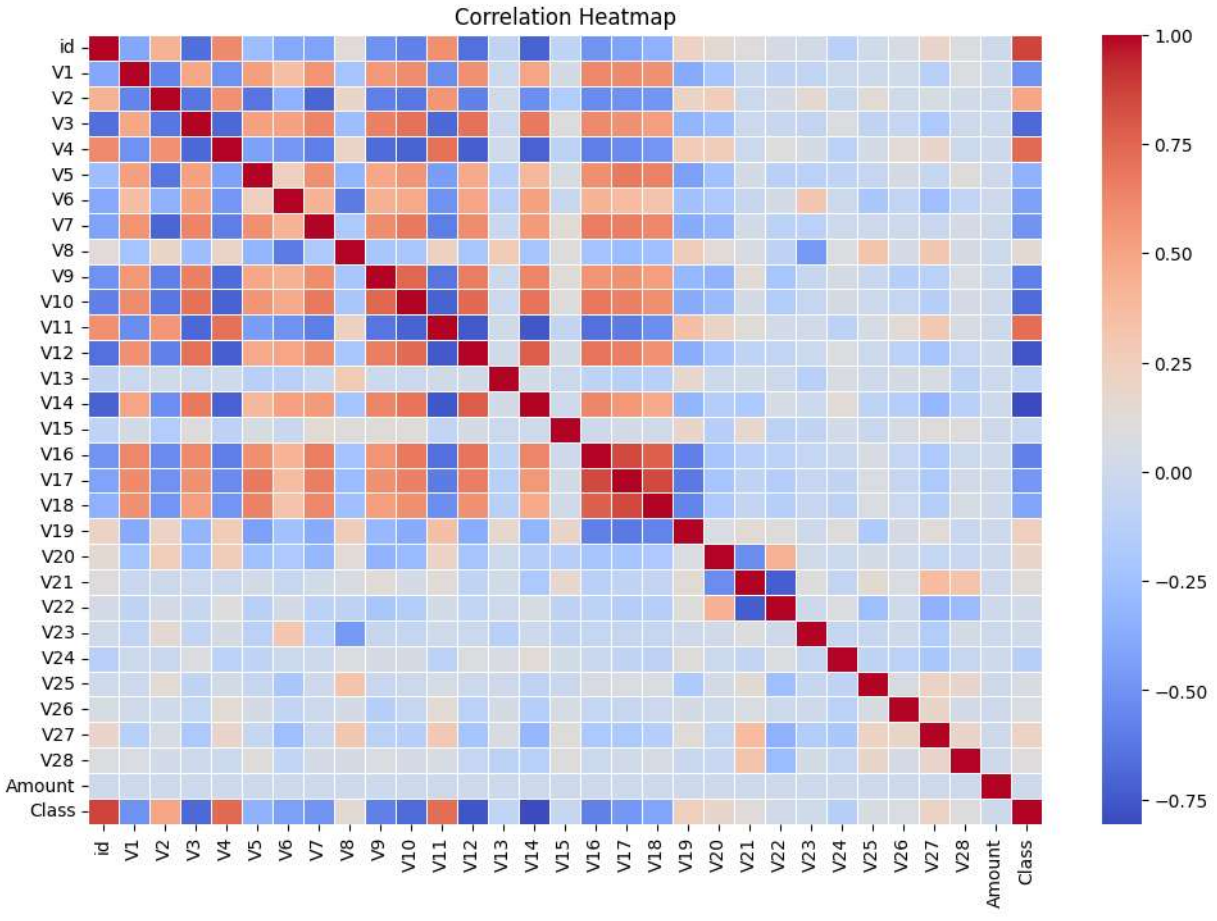
In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plot
import seaborn as sns

# Load the Dataset
df = pd.read_csv('creditcard_2023.csv')
```

In [ ]:
```
# Investigate Dataset - checking for missing values and duplicated values
df.isna().sum()
df.duplicated().any()
```

Out[ ]:
```
False
```

In [ ]:
```
# Create Correlation Heatmap
correlation_matrix = df.corr()
plot.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', linewidths=0.5)
plot.title('Correlation Heatmap')
plot.show()
```

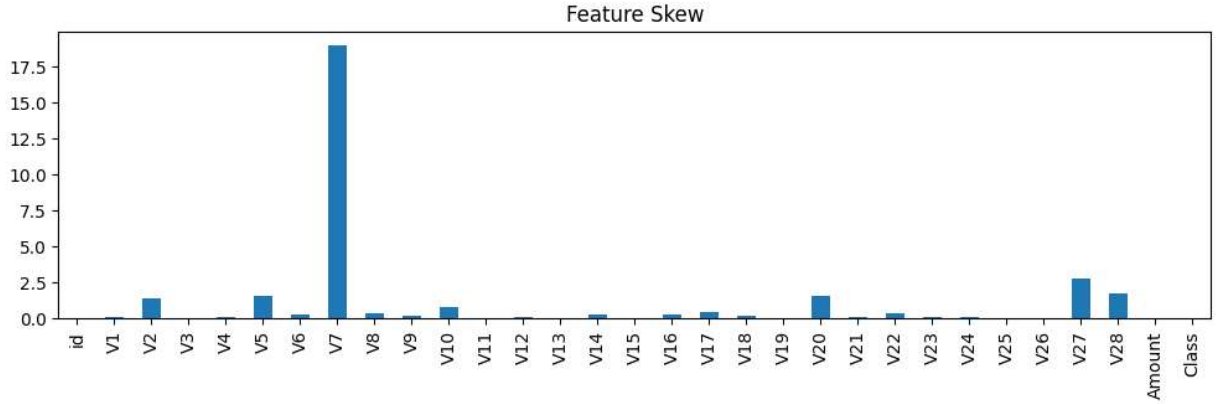


In [ ]:
```
# Skew Data
df.skew()
# V1, V10 and V23 is negatively skewed to a great extent
```

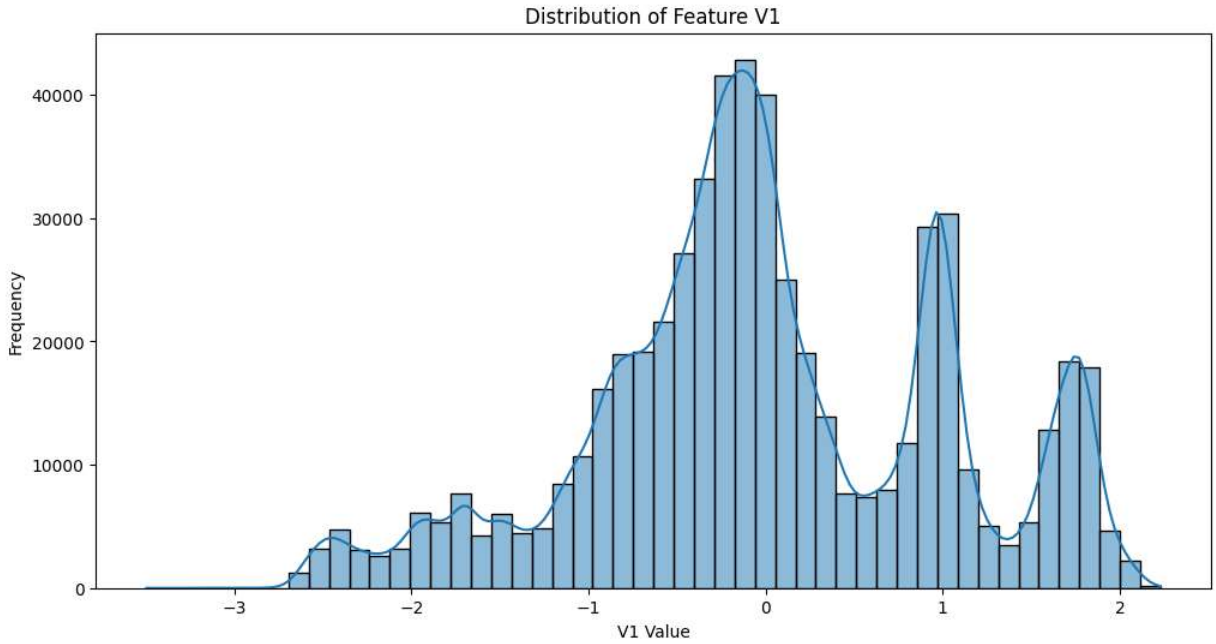
Out[ ]:
```
id       -6.579536e-16
V1       -8.341717e-02
V2       -1.397952e+00
V3        1.462221e-02
V4       -4.416893e-02
V5        1.506414e+00
V6       -2.016110e-01
V7        1.902687e+01
V8        2.999722e-01
V9        1.710575e-01
V10       7.404136e-01
V11      -2.089056e-02
V12       6.675895e-02
V13       1.490639e-02
V14       2.078348e-01
V15       1.123298e-02
V16       2.664070e-01
V17       3.730610e-01
V18       1.291911e-01
V19      -1.017123e-02
V20      -1.556460e+00
V21      -1.089833e-01
V22       3.185295e-01
V23      -9.968746e-02
V24       6.608974e-02
V25       2.300804e-02
V26      -1.895874e-02
V27       2.755452e+00
V28       1.724978e+00
Amount    1.655585e-03
Class     0.000000e+00
dtype: float64
```

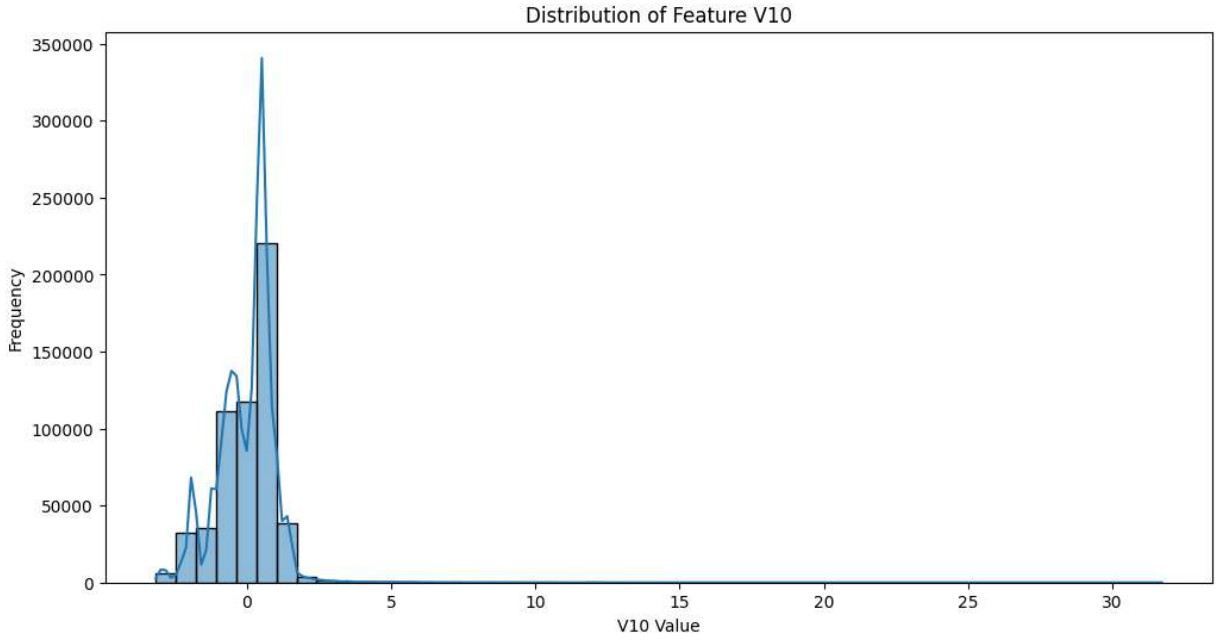
In [ ]:
```
from matplotlib.pyplot import figure
from matplotlib.pyplot import title

figure(figsize=(12,3))
features = df.columns.values.tolist()
skew = np.abs(df.skew())
feature_series = pd.Series(data=skew, index=features)
feature_series.plot.bar()
title('Feature Skew')
```

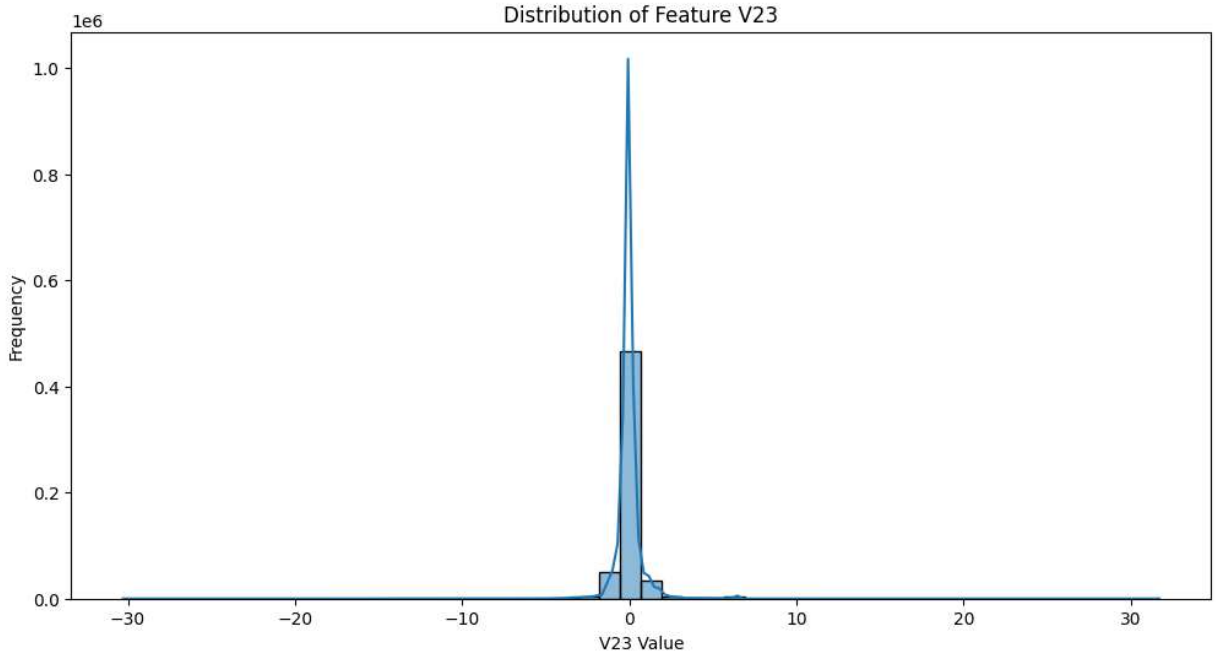
Out[ ]:
```
Text(0.5, 1.0, 'Feature Skew')
```



In [ ]:
```
# Create histogram for feature distribution of the negatively skewed values
plot.figure(figsize = (12, 6))
sns.histplot(df['V1'], bins = 50, kde = True)
plot.title('Distribution of Feature V1')
plot.xlabel('V1 Value')
plot.ylabel('Frequency')
plot.show()

plot.figure(figsize = (12, 6))
sns.histplot(df['V10'], bins = 50, kde = True)
plot.title('Distribution of Feature V10')
plot.xlabel('V10 Value')
plot.ylabel('Frequency')
plot.show()

plot.figure(figsize = (12, 6))
sns.histplot(df['V23'], bins = 50, kde = True)
plot.title('Distribution of Feature V23')
plot.xlabel('V23 Value')
plot.ylabel('Frequency')
plot.show()
```

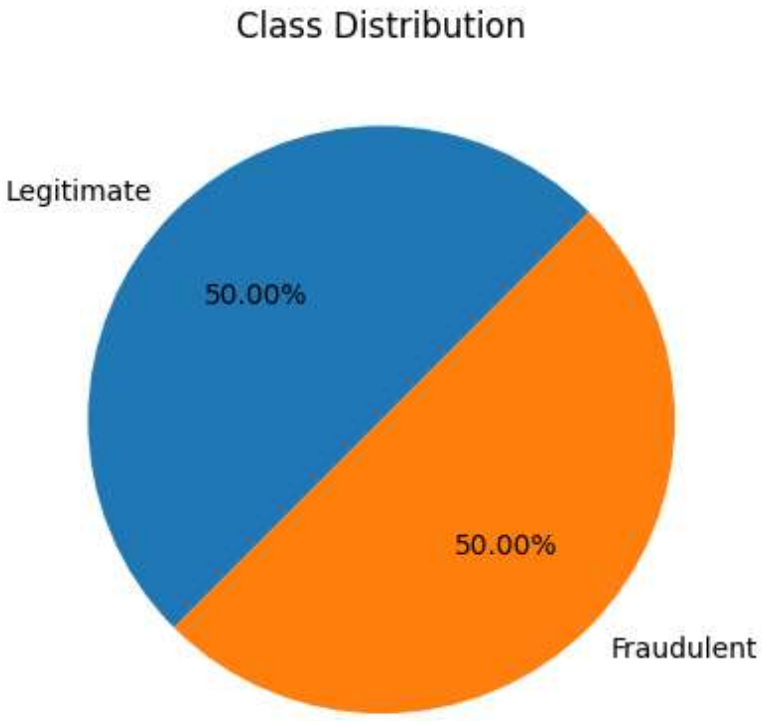






In [ ]:
```
features = df.drop(['id', 'Class'], axis=1)
labels = df['Class']
```

In [ ]:
```
# Class Distribution

unique, counts = np.unique(labels, return_counts=True)

legend = ['Legitimate', 'Fraudulent']

plot.pie(counts, autopct='%3.2f%%', labels=legend, startangle=45)
plot.title('Class Distribution')
plot.show()
```



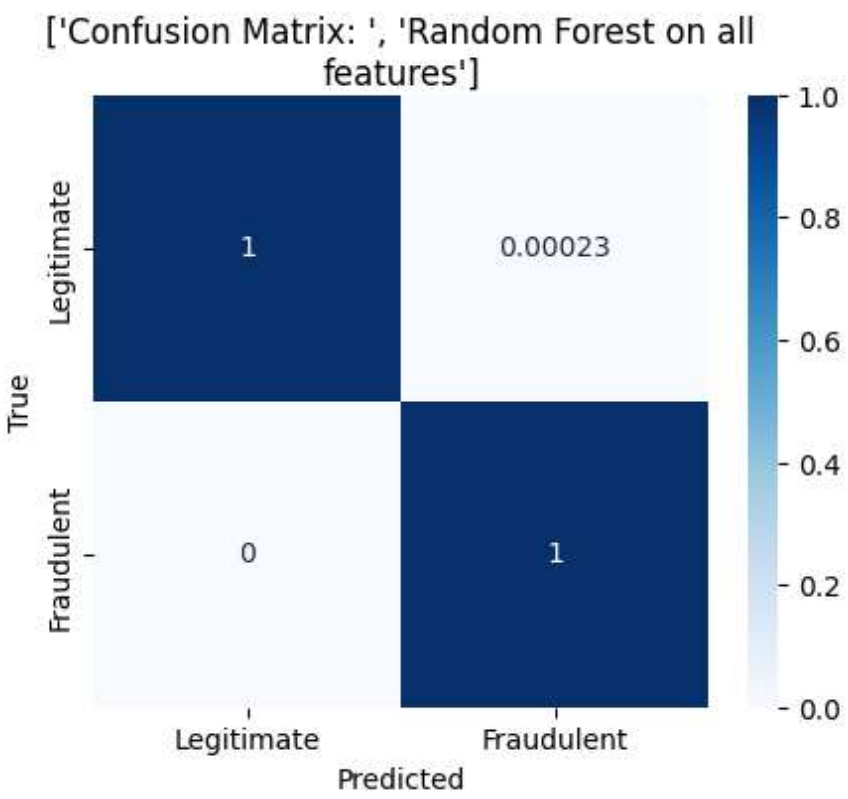
In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
```

In [ ]:
```
def train_random_forest(data_features, data_labels, title, max_tree_depth=None):

    # Description: This funtion takes in feature and label data to train a random fo
    # it prints out a classification report and confusion matrix to summarize the re
    # classifier training

    # Input:
    # data_features - features to train on
    # lables - corresponding classes to train on
    # title - string with description of data
    # max_tree_depth - max_depth of estimator decision tress. Default is max_depth=N

    # Output:
    # rf - random forest classifier trained on the input data
    # train_accuracy - training accuracy of the classifier
    # accuracy - testing accuracy of the calssifier

    X_train, X_test, y_train, y_test = train_test_split(data_features, data_labels,

    rf = RandomForestClassifier(class_weight='balanced', max_depth=max_tree_depth)
    rf.fit(X_train, y_train)

    y_pred = rf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    y_pred_train = rf.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_pred_train)
```

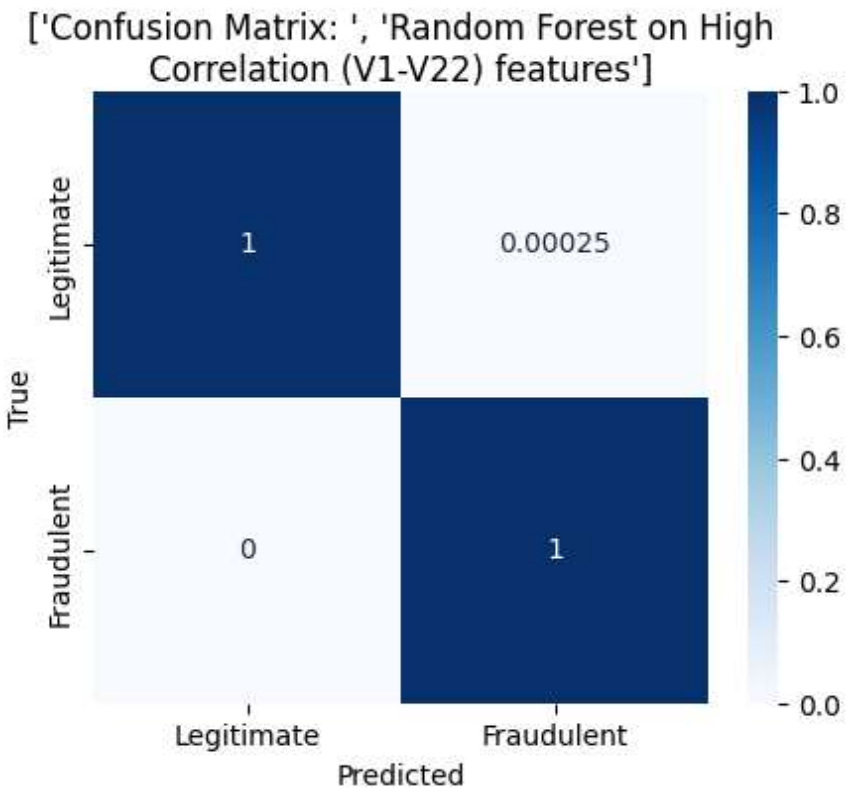
```
    print(title)
    print("Train Accuracy:", train_accuracy)
    print("Test Accuracy:", accuracy)

    print(classification_report(y_test, y_pred))

    confusion = confusion_matrix(y_test, y_pred)
    confusion = confusion / np.sum(confusion, axis=1)

    plot.figure(figsize=(5,4))
    sns.heatmap(confusion, annot=True, cmap=plot.cm.Blues)
    class_names = ['Legitimate', 'Fraudulent']
    tick_marks = np.arange(len(class_names)) + 0.5
    plot.xticks(tick_marks, class_names)
    plot.yticks(tick_marks, class_names)
    plot.xlabel('Predicted')
    plot.ylabel('True')
    plot.title(['Confusion Matrix: ', title], wrap=True)
    plot.show()

    return rf, train_accuracy, accuracy, X_test
```

In [ ]:
```
rf_all, train_acc_all, test_acc_all, X_test_all = train_random_forest(features, labe

hi_corr_features = features.iloc[:, 0:21]
rf_hi_corr, train_acc_hi_corr, test_acc_hi_corr, X_test_hi_corr = train_random_fores
```

```
Random Forest on all features
Train Accuracy: 1.0
Test Accuracy: 0.9998856901675958
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     57027
           1       1.00      1.00      1.00     56699

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```




```
Random Forest on High Correlation (V1-V22) features
Train Accuracy: 1.0
Test Accuracy: 0.9998768971035648
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56609
           1       1.00      1.00      1.00     57117

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```



In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2)

rf = RandomForestClassifier(class_weight='balanced')
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

y_pred_train = rf.predict(X_train)
train_accuracy = accuracy_score(y_train, y_pred_train)

print("Random Forest on all features")
print("Train Accuracy:", train_accuracy)
print("Validation Accuracy:", accuracy)

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

```
Random Forest on all features
Train Accuracy: 1.0
Validation Accuracy: 0.9998681040395336
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56529
           1       1.00      1.00      1.00     57197

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

In [ ]:
```
X_train2, X_test2, y_train2, y_test2 = train_test_split(features.iloc[:, 0:21], labe
```

```
rf2 = RandomForestClassifier(class_weight='balanced')
rf2.fit(X_train2, y_train2)

y_pred2 = rf2.predict(X_test2)
accuracy2 = accuracy_score(y_test2, y_pred2)

y_pred_train2 = rf2.predict(X_train2)
train_accuracy2 = accuracy_score(y_train2, y_pred_train2)

print("Random Forest on High Correlation (V1-V22) features")
print("Train Accuracy:", train_accuracy2)
print("Validation Accuracy:", accuracy2)

print(classification_report(y_test2, y_pred2))
```

```
Random Forest on High Correlation (V1-V22) features
Train Accuracy: 1.0
Validation Accuracy: 0.9998944832316269
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56865
           1       1.00      1.00      1.00     56861

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

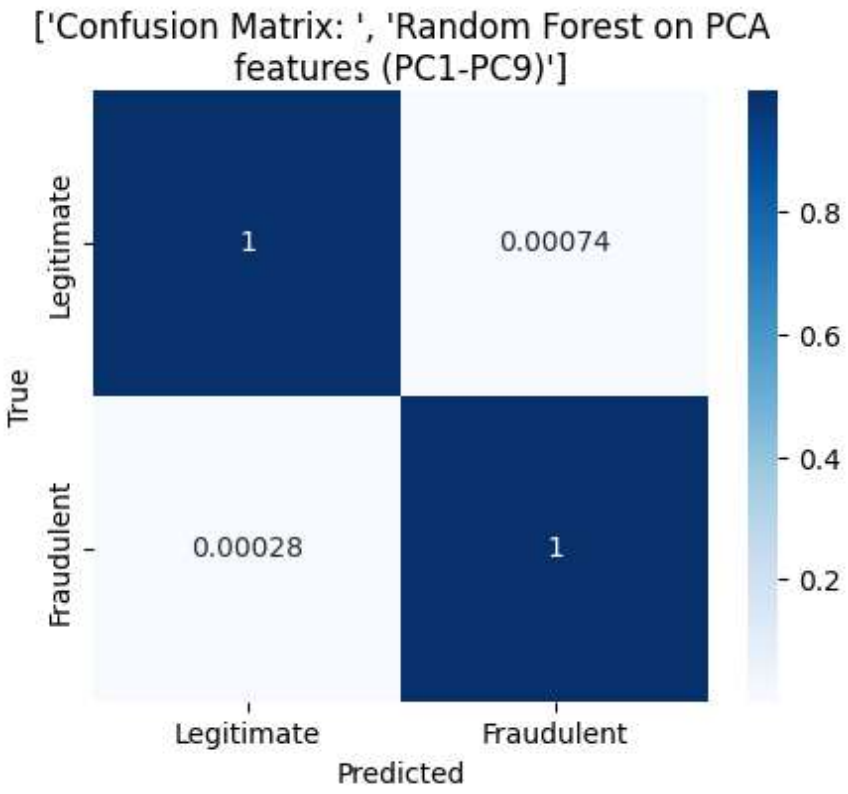
In [ ]:
```
#Look at cumulative explained variance to select number of PCA components (We chose
from sklearn.decomposition import PCA

n_features = 21
for i in range(2, n_features):
  pca = PCA(n_components=i)
  pca.fit(features.iloc[:, 0:21])
  print(i, ': ', np.sum(pca.explained_variance_ratio_))
```

```
2 :  0.5515749428005279
3 :  0.6231340737442913
4 :  0.688134480725721
5 :  0.7360301946682688
6 :  0.7728009538721754
7 :  0.8049423038874883
8 :  0.8291614550955293
9 :  0.85094332822088
10 :  0.8708689716485893
11 :  0.8899439656629095
12 :  0.9055681709953399
13 :  0.9195045120271931
14 :  0.9329460534709804
15 :  0.945092875506545
16 :  0.9568498911683869
17 :  0.967347429605479
18 :  0.9768002216856297
19 :  0.9859213615414328
20 :  0.9947349089755183
```

In [ ]:
```
pca = PCA(n_components=9)
pca.fit(features.iloc[:, 0:21])
pca_features = pca.transform(features.iloc[:, 0:n_features])

# print('PCA Components: \n', pca.components_)
# print('PCA Features: \n', pca_features)
```

In [ ]:
```
rf_pca, train_acc_pca, test_acc_pca, X_test_pca = train_random_forest(pca_features,
```

```
Random Forest on PCA features (PC1-PC9)
Train Accuracy: 1.0
Test Accuracy: 0.9994900022861967
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56905
           1       1.00      1.00      1.00     56821

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```



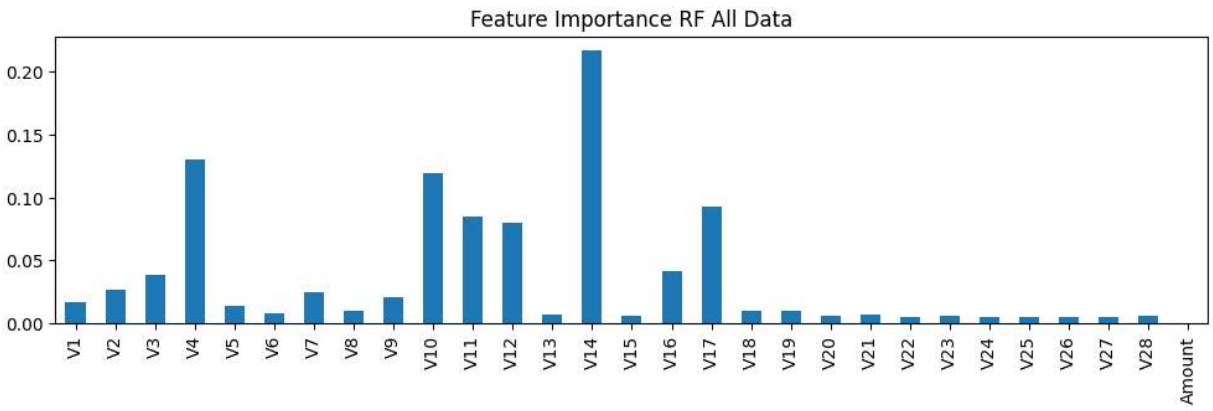
In [ ]:
```
X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(pca_features, la

rf_pca = RandomForestClassifier(class_weight='balanced')
rf_pca.fit(X_train_pca, y_train_pca)


y_pred_pca = rf_pca.predict(X_test_pca)
accuracy_pca = accuracy_score(y_test_pca, y_pred_pca)

y_pred_train_pca = rf_pca.predict(X_train_pca)
```

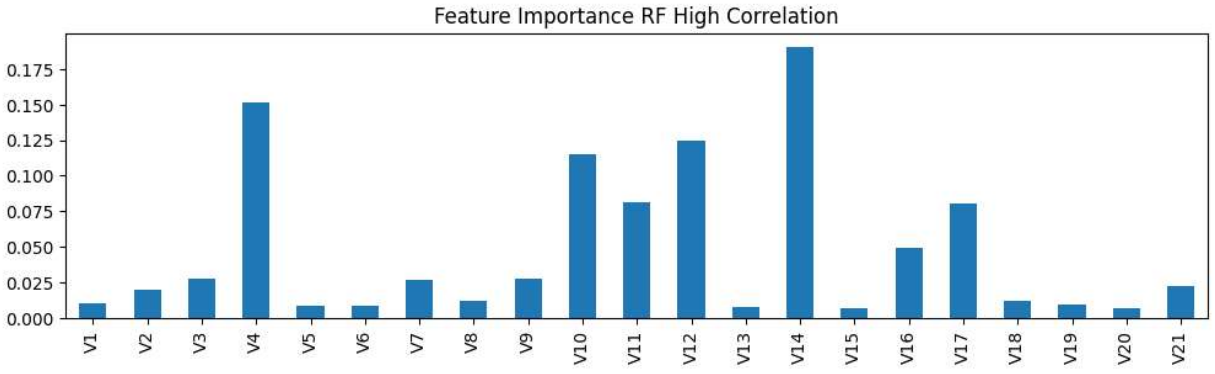
```
train_accuracy_pca = accuracy_score(y_train_pca, y_pred_train_pca)

print("Random Forest on PCA features (PC1-PC9)")
print("Train Accuracy:", train_accuracy_pca)
print("Validation Accuracy:", accuracy_pca)

print(classification_report(y_test_pca, y_pred_pca))
```

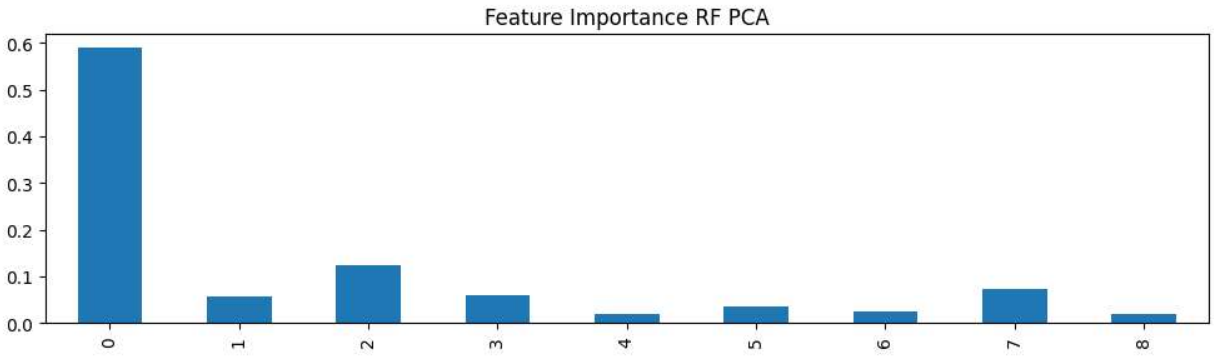
```
Random Forest on PCA features (PC1-PC9)
Train Accuracy: 1.0
Validation Accuracy: 0.99943724390201
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56568
           1       1.00      1.00      1.00     57158

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

In [ ]:
```
figure(figsize=(12,3))
feat = X_test_all.columns.values.tolist()
importance = rf_all.feature_importances_.tolist()
feature_series = pd.Series(data=importance, index=feat)
feature_series.plot.bar()
title('Feature Importance RF All Data')

figure(figsize=(12,3))
feat = X_test_hi_corr.columns.values.tolist()
importance = rf_hi_corr.feature_importances_.tolist()
feature_series = pd.Series(data=importance, index=feat)
feature_series.plot.bar()
title('Feature Importance RF High Correlation')

figure(figsize=(12,3))
feat = pd.DataFrame(X_test_pca).columns.values.tolist()
importance = rf_pca.feature_importances_.tolist()
feature_series = pd.Series(data=importance, index=feat)
feature_series.plot.bar()
title('Feature Importance RF PCA')
```

Out[ ]:
```
Text(0.5, 1.0, 'Feature Importance RF PCA')
```

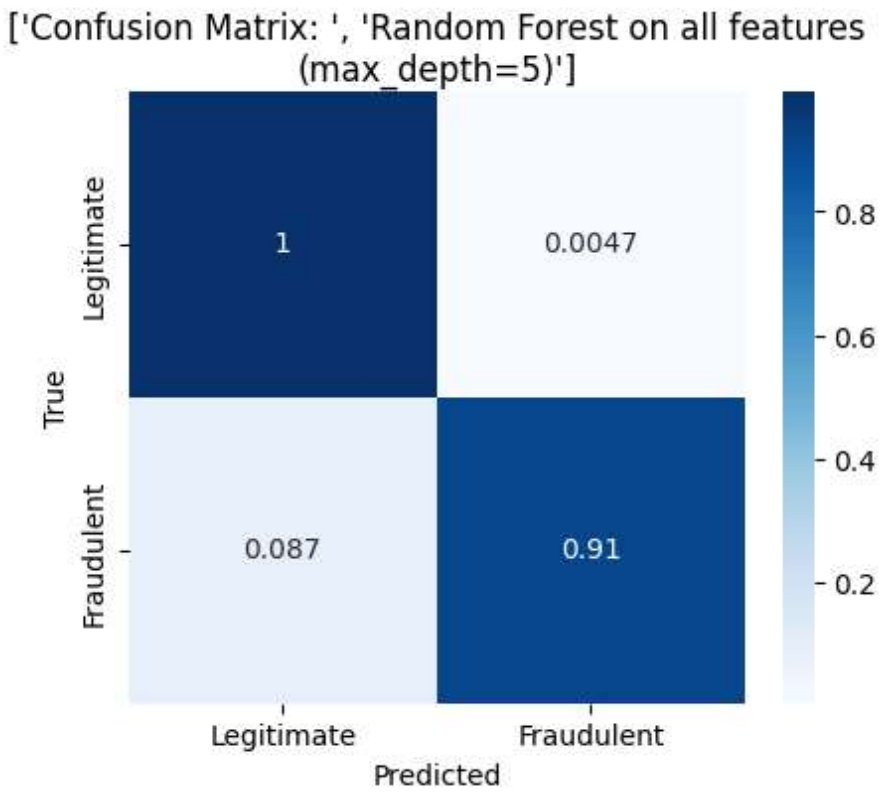






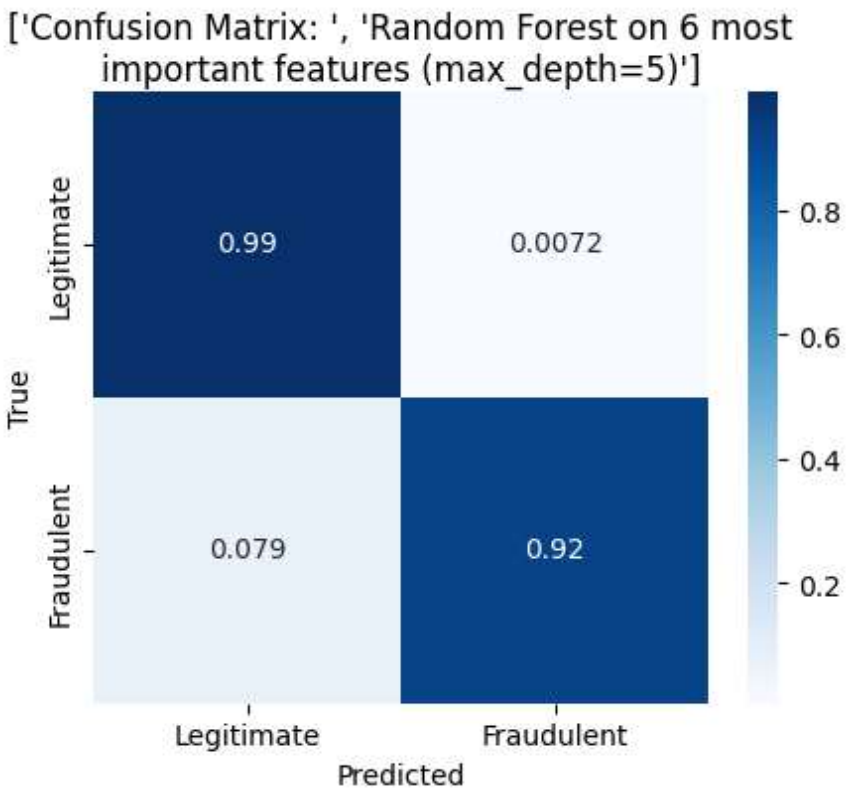
In [ ]:
```
rf_all_5d, train_acc_all_5d, test_acc_all_5d, X_test_acc_all_5d = train_random_fores
```

```
Random Forest on all features (max_depth=5)
Train Accuracy: 0.9547948578161546
Test Accuracy: 0.954258480910258
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     56970
           1       0.99      0.91      0.95     56756

    accuracy                           0.95    113726
   macro avg       0.96      0.95      0.95    113726
weighted avg       0.96      0.95      0.95    113726
```

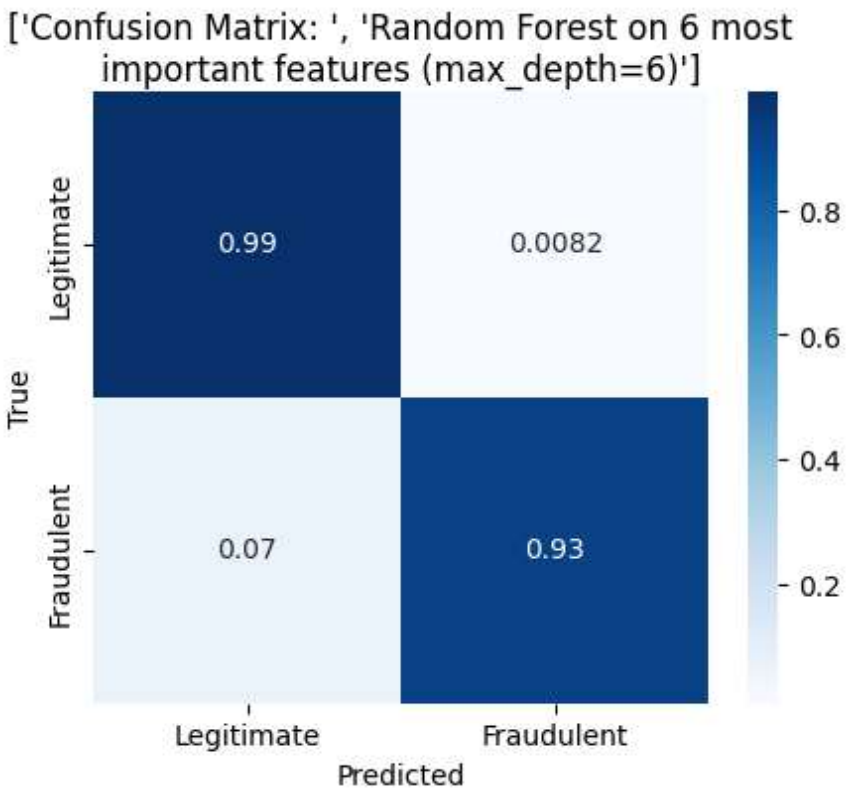


In [ ]:
```
important_features_6 = df[['V4', 'V11', 'V10', 'V12', 'V14', 'V17']]
rf_6_imp_5d, train_acc_6_imp_5d, test_acc_6_imp_5d, X_test_acc_6_imp_5d = train_rand
rf_6_imp_6d, train_acc_6_imp_6d, test_acc_6_imp_6d, X_test_acc_6_imp_6d = train_rand
rf_6_imp_7d, train_acc_6_imp_7d, test_acc_6_imp_7d, X_test_acc_6_imp_7d = train_rand
rf_6_imp_8d, train_acc_6_imp_8d, test_acc_6_imp_8d, X_test_acc_6_imp_8d = train_rand
rf_6_imp_8d, train_acc_6_imp_9d, test_acc_6_imp_9d, X_test_acc_6_imp_9d = train_rand
rf_6_imp_8d, train_acc_6_imp_10d, test_acc_6_imp_10d, X_test_acc_6_imp_10d = train_r
```

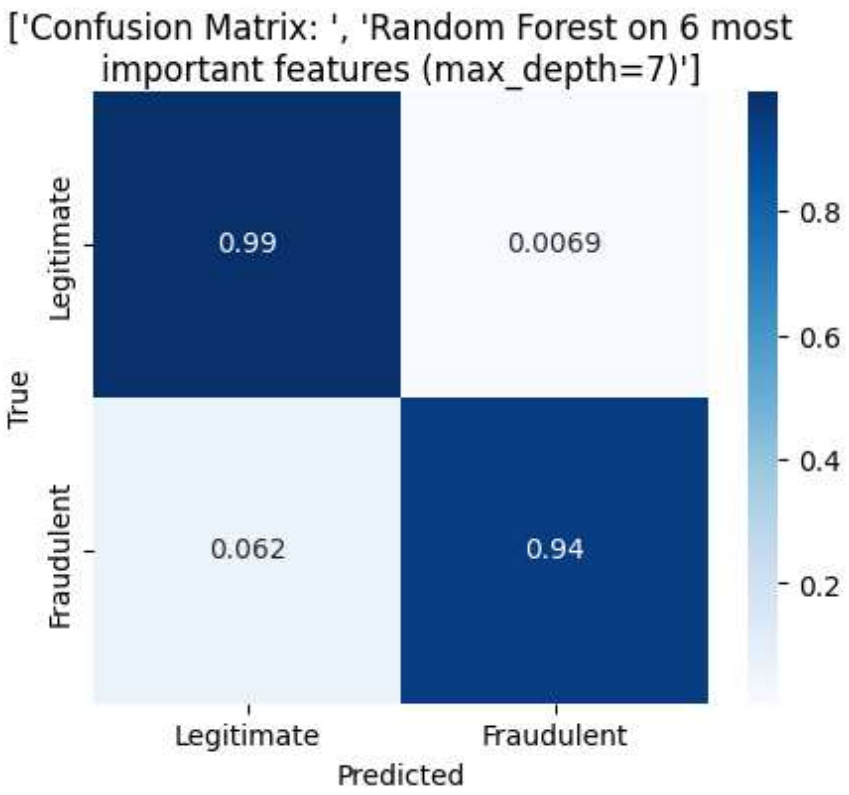
```
Random Forest on 6 most important features (max_depth=5)
Train Accuracy: 0.9572195452227283
Test Accuracy: 0.9569667446318344
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     56509
           1       0.99      0.92      0.96     57217

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```



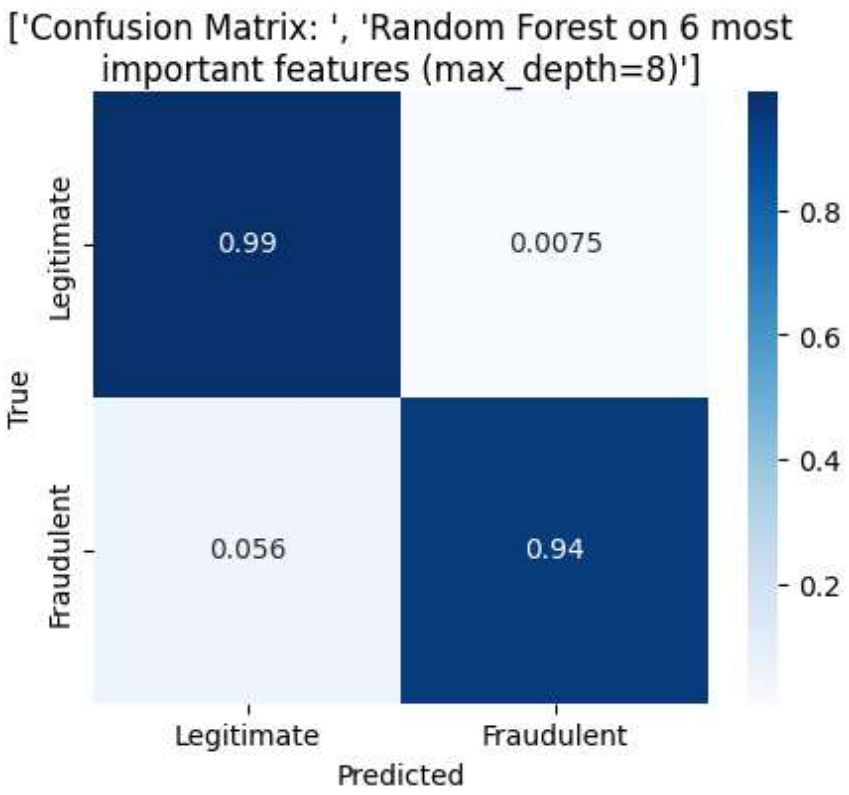

```
Random Forest on 6 most important features (max_depth=6)
Train Accuracy: 0.961446810755676
Test Accuracy: 0.9607125899090797
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     56747
           1       0.99      0.93      0.96     56979

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```



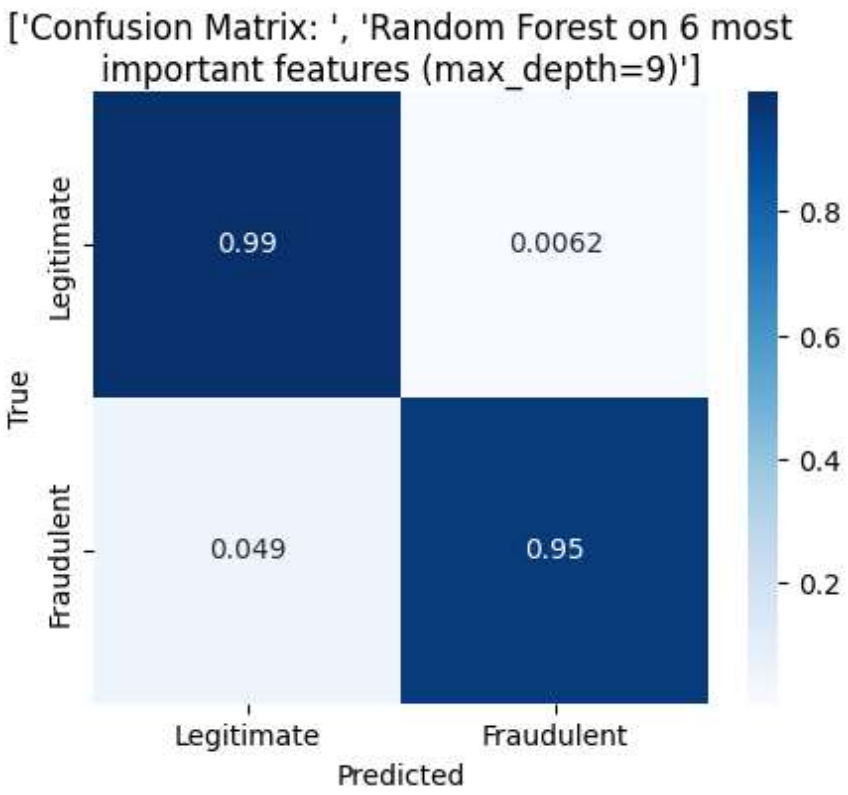

```
Random Forest on 6 most important features (max_depth=7)
Train Accuracy: 0.9653773103775741
Test Accuracy: 0.9653904999736208
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     57149
           1       0.99      0.94      0.96     56577

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726
```



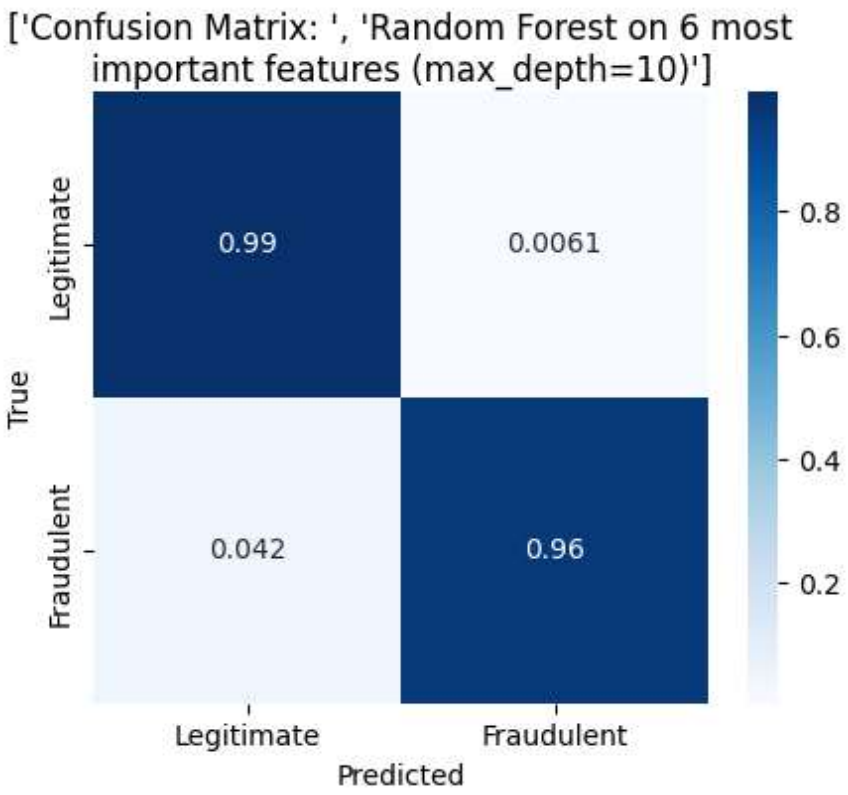

```
Random Forest on 6 most important features (max_depth=8)
Train Accuracy: 0.9688835446599722
Test Accuracy: 0.9684065209362854
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     56874
           1       0.99      0.94      0.97     56852

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726
```




```
Random Forest on 6 most important features (max_depth=9)
Train Accuracy: 0.9726975361834584
Test Accuracy: 0.9723106413660905
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     57040
           1       0.99      0.95      0.97     56686

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726
```

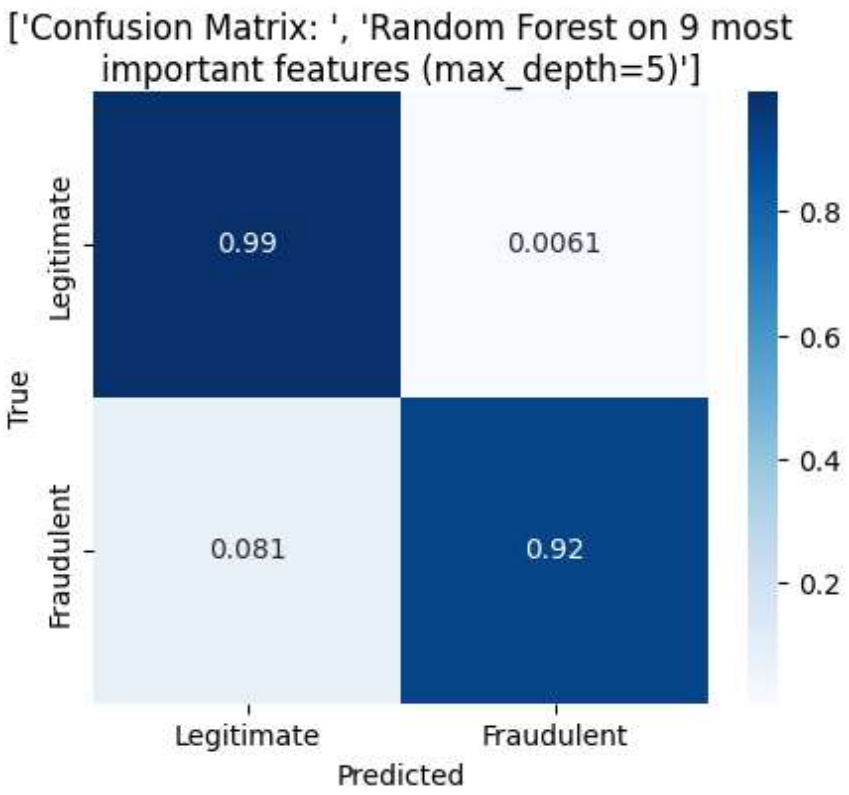



```
Random Forest on 6 most important features (max_depth=10)
Train Accuracy: 0.9771314387211367
Test Accuracy: 0.9756871779540298
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     57060
           1       0.99      0.96      0.98     56666

    accuracy                           0.98    113726
   macro avg       0.98      0.98      0.98    113726
weighted avg       0.98      0.98      0.98    113726
```

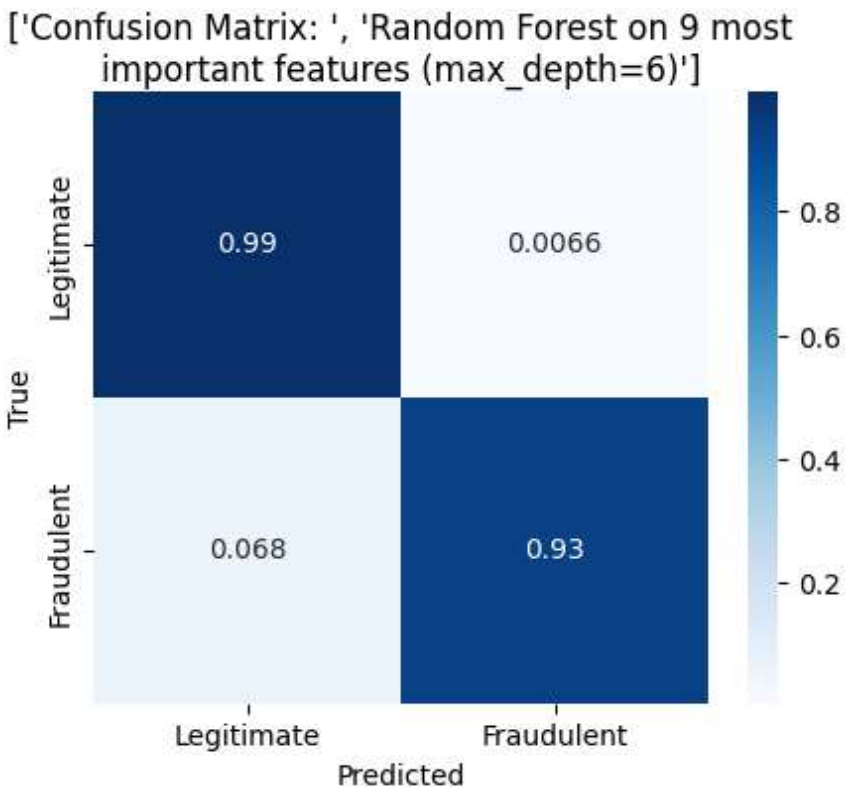


In [ ]:
```
important_features_9 = df[['V4', 'V10', 'V12', 'V14', 'V17', 'V11', 'V3', 'V16', 'V2
rf_9_imp_5d, train_acc_9_imp_5d, test_acc_9_imp_5d, X_test_acc_9_imp_5d = train_rand
rf_9_imp_6d, train_acc_9_imp_6d, test_acc_9_imp_6d, X_test_acc_9_imp_6d = train_rand
rf_9_imp_7d, train_acc_9_imp_7d, test_acc_9_imp_7d, X_test_acc_9_imp_7d = train_rand
rf_9_imp_8d, train_acc_9_imp_8d, test_acc_9_imp_8d, X_test_acc_9_imp_8d = train_rand
rf_9_imp_9d, train_acc_9_imp_9d, test_acc_9_imp_9d, X_test_acc_9_imp_9d = train_rand
rf_9_imp_10d, train_acc_9_imp_10d, test_acc_9_imp_10d, X_test_acc_9_imp_10d = train_
```

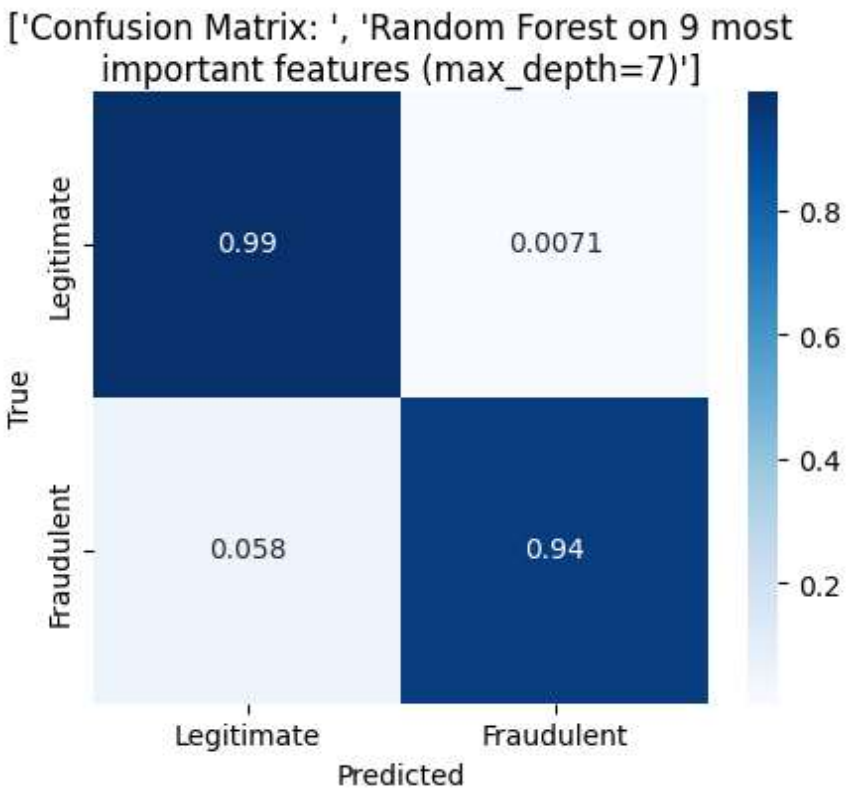
```
Random Forest on 9 most important features (max_depth=5)
Train Accuracy: 0.956311661361518
Test Accuracy: 0.9564831261101243
              precision    recall  f1-score   support

           0       0.92      0.99      0.96     56769
           1       0.99      0.92      0.95     56957

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```



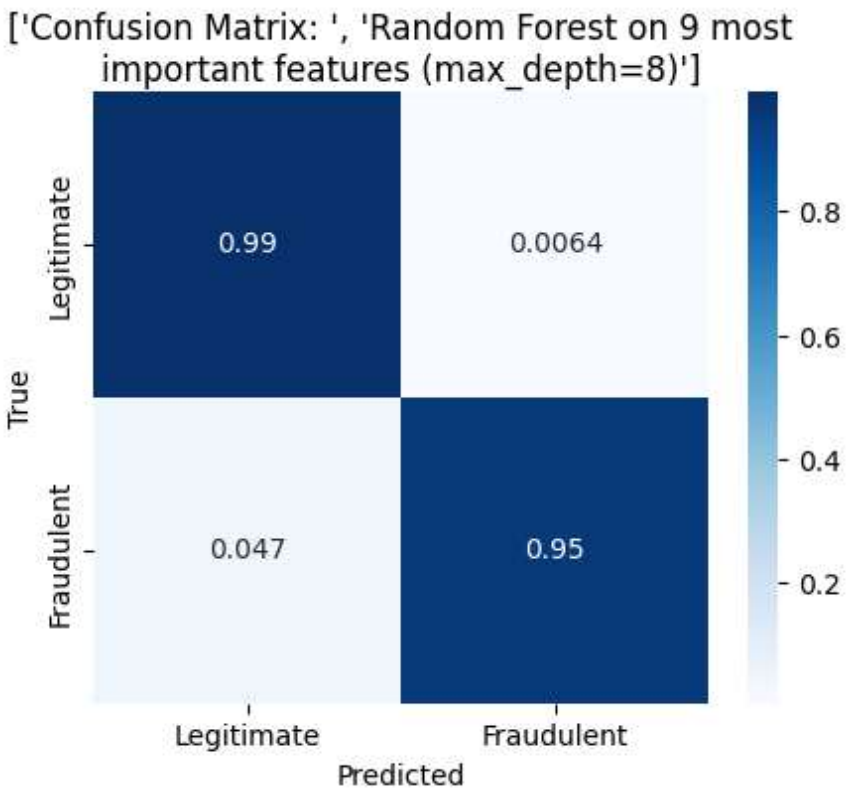

```
Random Forest on 9 most important features (max_depth=6)
Train Accuracy: 0.9620974974939768
Test Accuracy: 0.9625943056117334
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     57017
           1       0.99      0.93      0.96     56709

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```



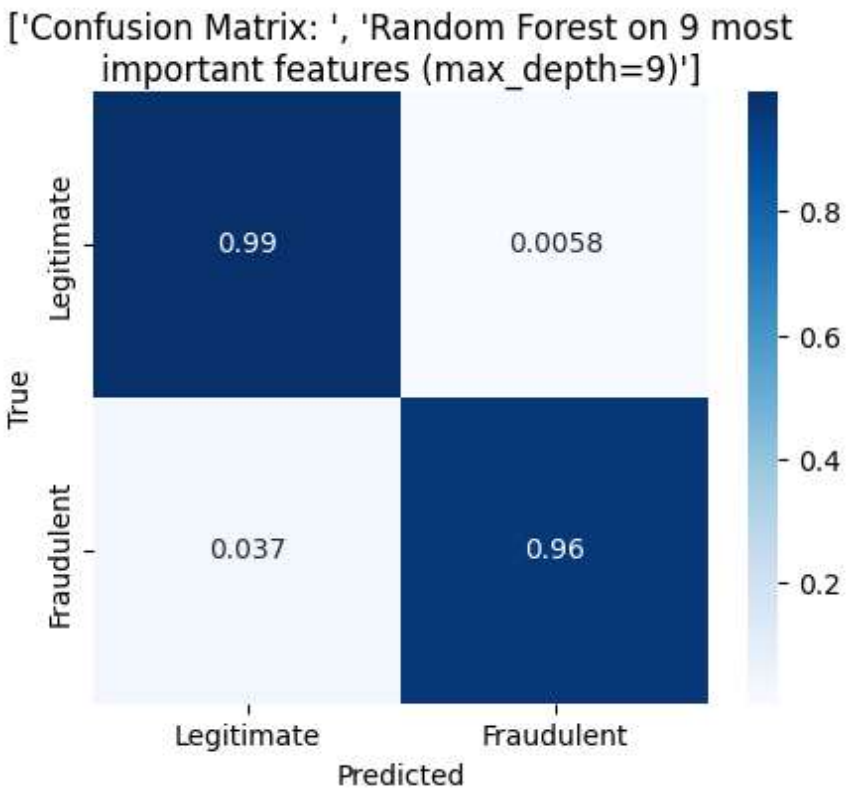

```
Random Forest on 9 most important features (max_depth=7)
Train Accuracy: 0.969098974728734
Test Accuracy: 0.9673249740604611
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     57069
           1       0.99      0.94      0.97     56657

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726
```



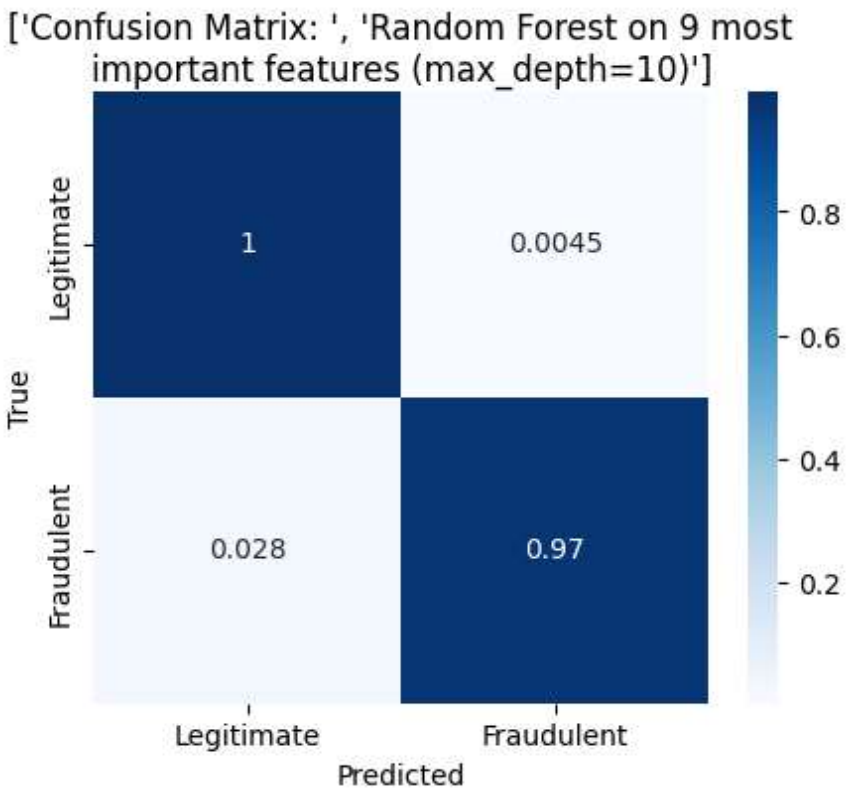

```
Random Forest on 9 most important features (max_depth=8)
Train Accuracy: 0.9741901588027364
Test Accuracy: 0.9733658090498215
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     56865
           1       0.99      0.95      0.97     56861

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726
```




```
Random Forest on 9 most important features (max_depth=9)
Train Accuracy: 0.9787251815767722
Test Accuracy: 0.9786680266605702
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     56941
           1       0.99      0.96      0.98     56785

    accuracy                           0.98    113726
   macro avg       0.98      0.98      0.98    113726
weighted avg       0.98      0.98      0.98    113726
```

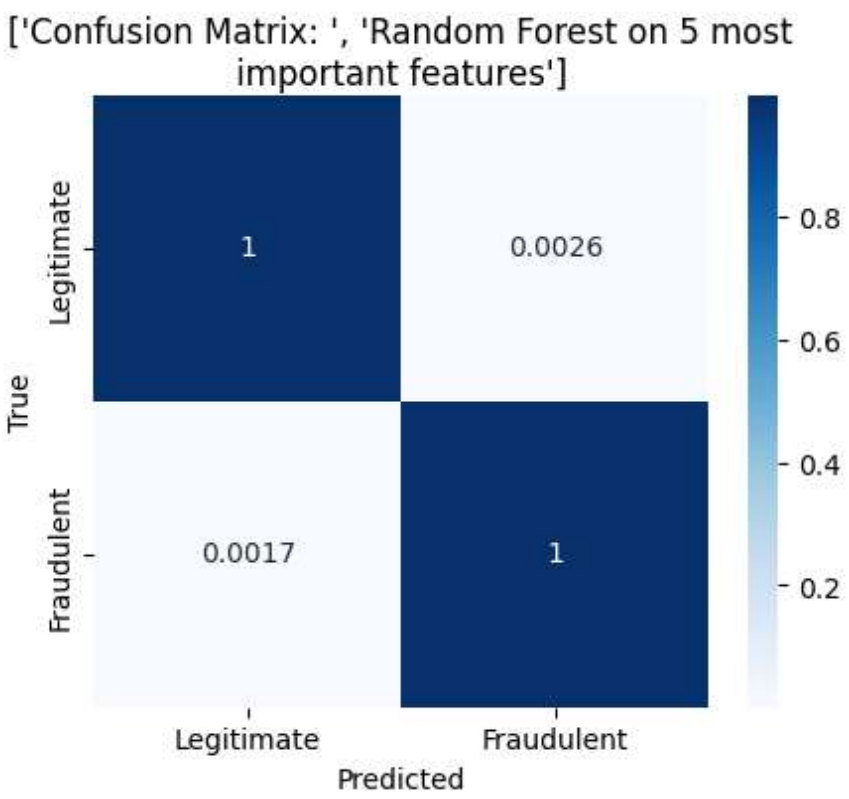



```
Random Forest on 9 most important features (max_depth=10)
Train Accuracy: 0.9833305488630568
Test Accuracy: 0.9837504176705415
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     56515
           1       1.00      0.97      0.98     57211

    accuracy                           0.98    113726
   macro avg       0.98      0.98      0.98    113726
weighted avg       0.98      0.98      0.98    113726
```

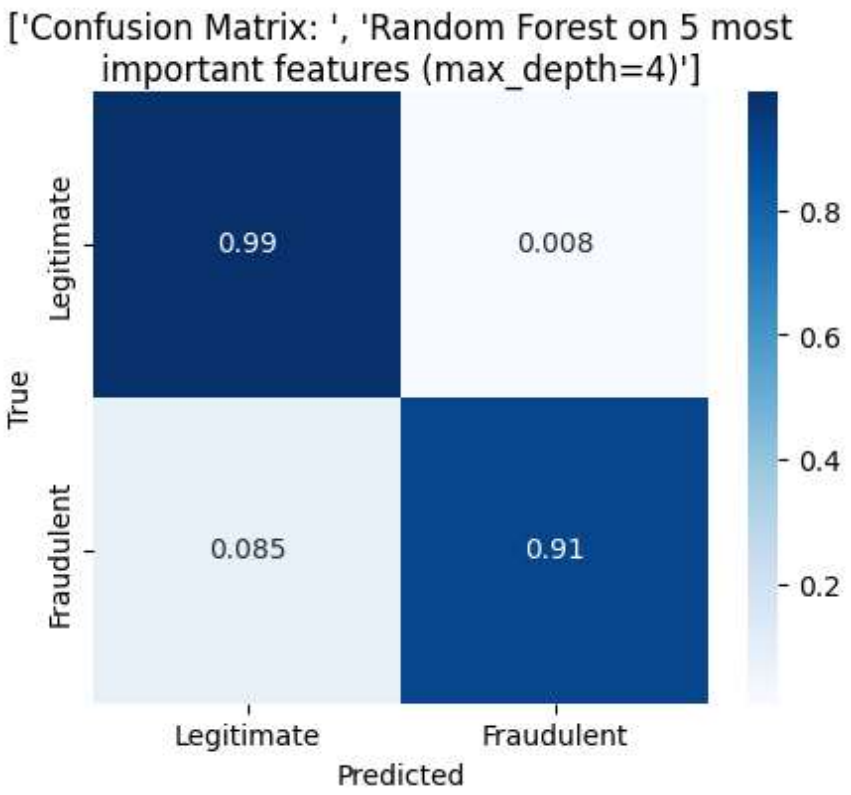


In [ ]:
```
important_features = df[['V4', 'V10', 'V12', 'V14', 'V17']]
rf_5_imp, train_acc_5_imp, test_acc_5_imp, X_test_acc_5_imp = train_random_forest(im

rf_5_imp_4d, train_acc_5_imp_4d, test_acc_5_imp_4d, X_test_acc_5_imp_4d = train_rand
rf_5_imp_5d, train_acc_5_imp_5d, test_acc_5_imp_5d, X_test_acc_5_imp_5d = train_rand
rf_5_imp_6d, train_acc_5_imp_6d, test_acc_5_imp_6d, X_test_acc_5_imp_6d = train_rand
rf_5_imp_7d, train_acc_5_imp_7d, test_acc_5_imp_7d, X_test_acc_5_imp_7d = train_rand
```

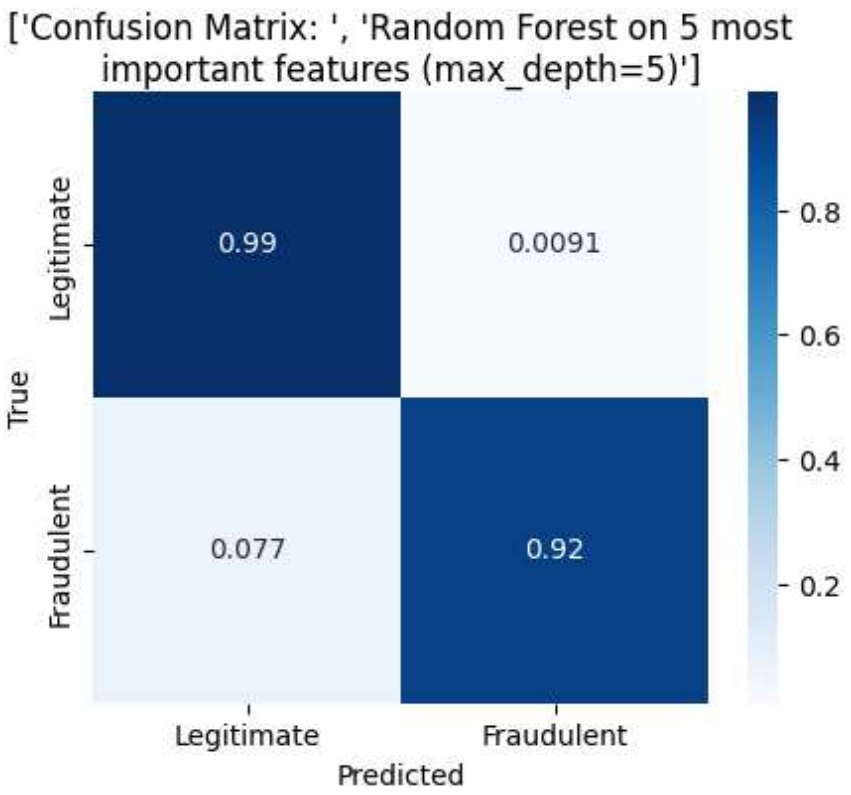
```
Random Forest on 5 most important features
Train Accuracy: 1.0
Test Accuracy: 0.9978369062483513
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56882
           1       1.00      1.00      1.00     56844

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```



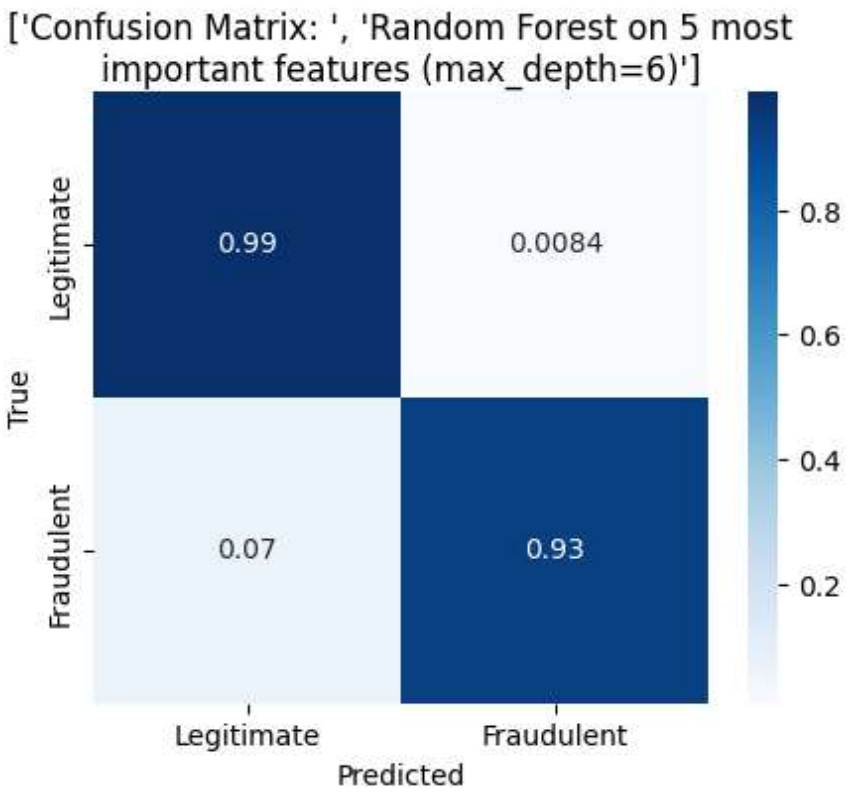

```
Random Forest on 5 most important features (max_depth=4)
Train Accuracy: 0.9538254225067266
Test Accuracy: 0.9532472785466823
              precision    recall  f1-score   support

           0       0.92      0.99      0.96     56894
           1       0.99      0.91      0.95     56832

    accuracy                           0.95    113726
   macro avg       0.96      0.95      0.95    113726
weighted avg       0.96      0.95      0.95    113726
```



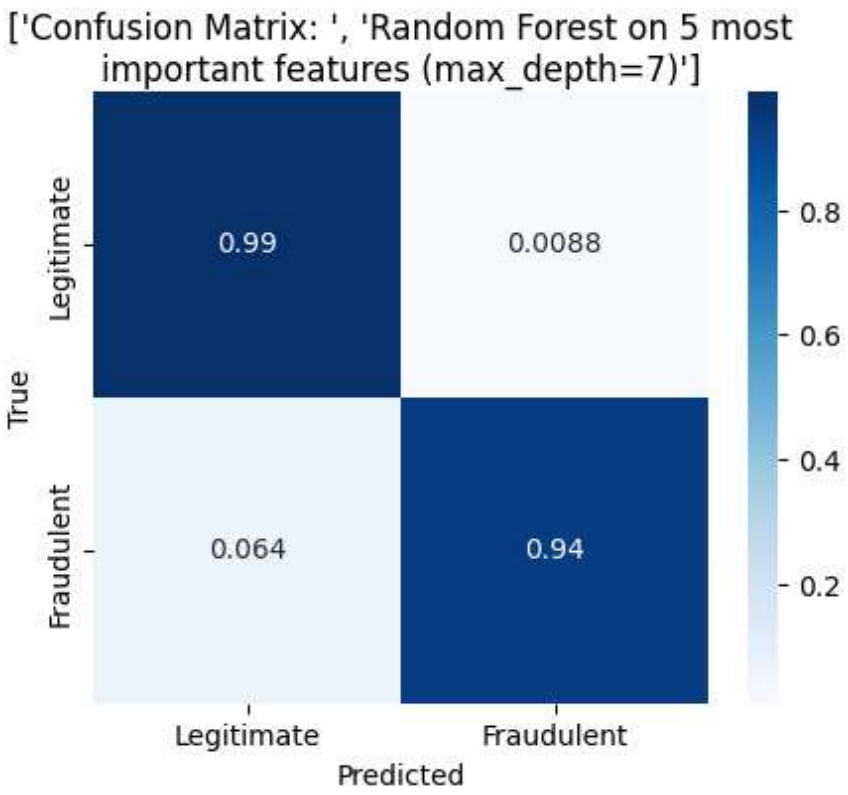

```
Random Forest on 5 most important features (max_depth=5)
Train Accuracy: 0.9564413590559766
Test Accuracy: 0.95693157237571
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     56994
           1       0.99      0.92      0.96     56732

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```




```
Random Forest on 5 most important features (max_depth=6)
Train Accuracy: 0.9605147459683802
Test Accuracy: 0.9609675887659814
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     56702
           1       0.99      0.93      0.96     57024

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```

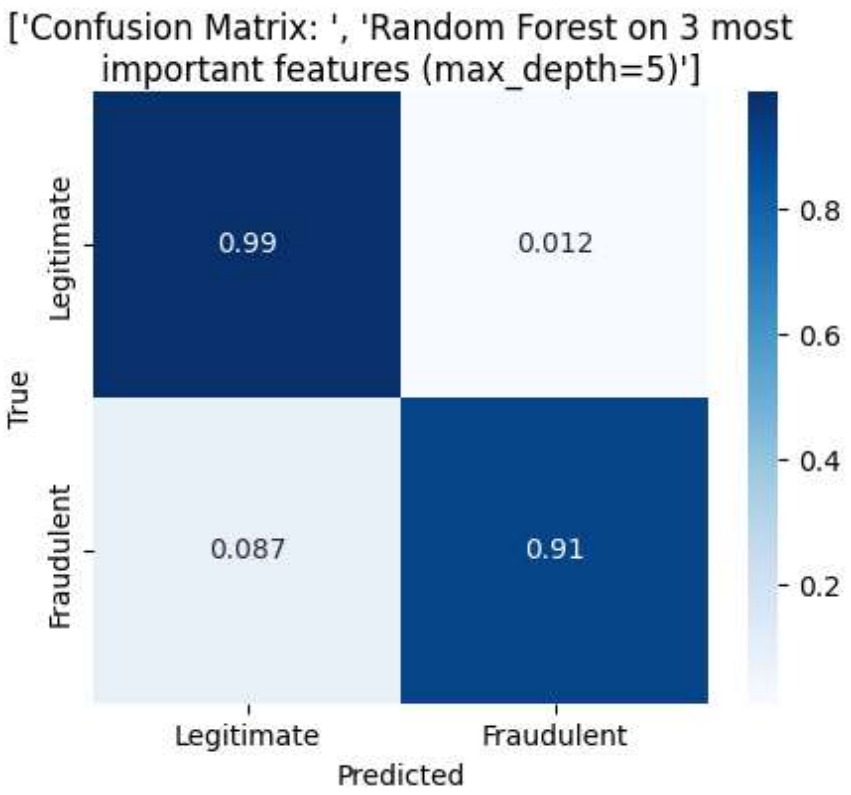



```
Random Forest on 5 most important features (max_depth=7)
Train Accuracy: 0.963838524172133
Test Accuracy: 0.96369343861562
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     56776
           1       0.99      0.94      0.96     56950

    accuracy                           0.96    113726
   macro avg       0.97      0.96      0.96    113726
weighted avg       0.97      0.96      0.96    113726
```



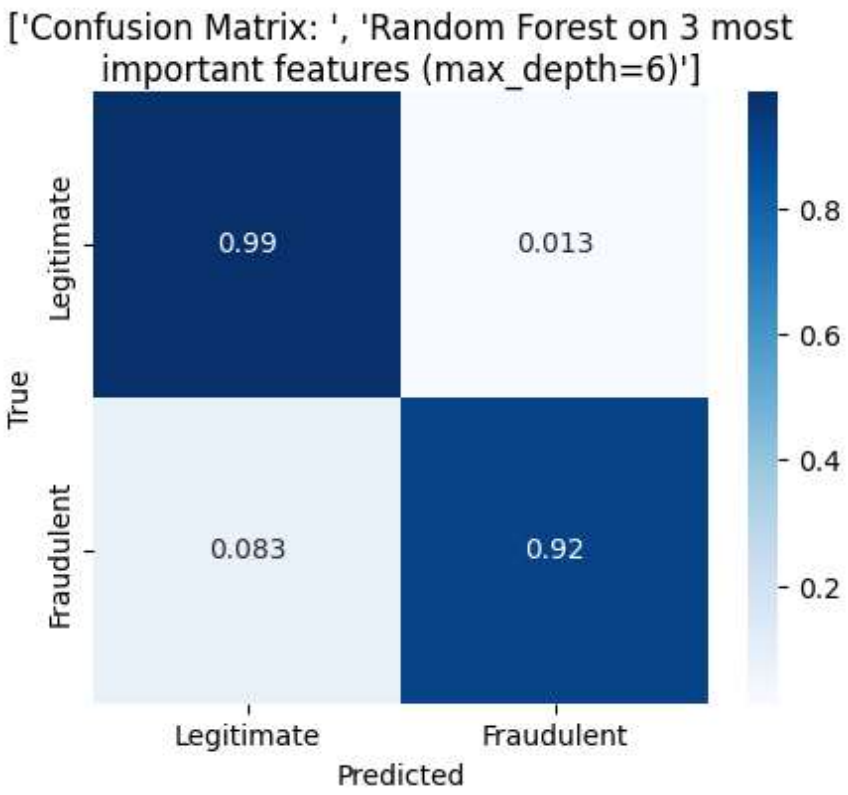
In [ ]:
```
important_features_3 = df[['V4', 'V10', 'V14']]
rf_3_imp_5d, train_acc_3_imp_5d, test_acc_3_imp_5d, X_test_acc_3_imp_5d = train_rand
```

```
Random Forest on 3 most important features (max_depth=5)
Train Accuracy: 0.949996043121186
Test Accuracy: 0.9503279812883597
              precision    recall  f1-score   support

           0       0.92      0.99      0.95     56976
           1       0.99      0.91      0.95     56750

    accuracy                           0.95    113726
   macro avg       0.95      0.95      0.95    113726
weighted avg       0.95      0.95      0.95    113726
```

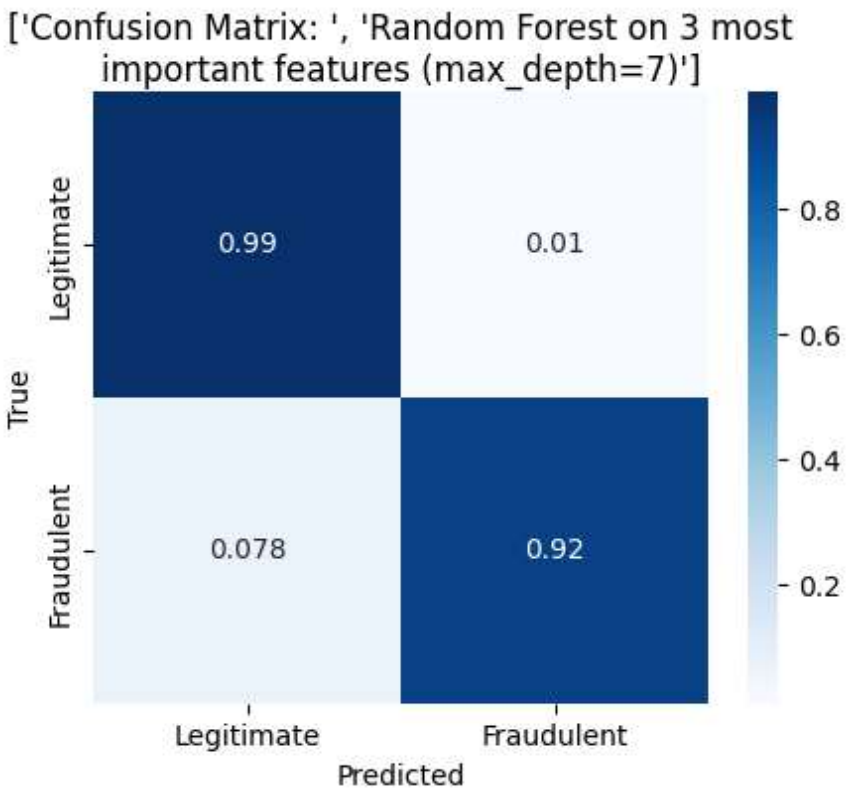


In [ ]:
```
rf_3_imp_6d, train_acc_3_imp_6d, test_acc_3_imp_6d, X_test_acc_3_imp_6d = train_rand
rf_3_imp_7d, train_acc_3_imp_7d, test_acc_3_imp_7d, X_test_acc_3_imp_7d = train_rand
rf_3_imp_8d, train_acc_3_imp_8d, test_acc_3_imp_8d, X_test_acc_3_imp_8d = train_rand
rf_3_imp_9d, train_acc_3_imp_9d, test_acc_3_imp_9d, X_test_acc_3_imp_9d = train_rand
rf_3_imp_10d, train_acc_3_imp_10d, test_acc_3_imp_10d, X_test_acc_3_imp_10d = train_
```

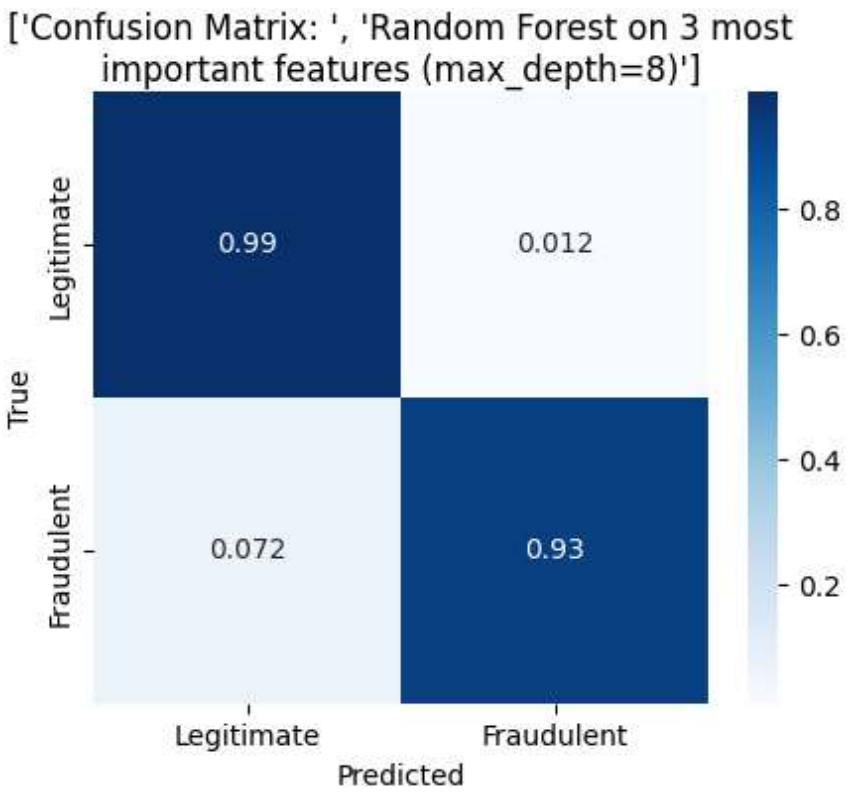
```
Random Forest on 3 most important features (max_depth=6)
Train Accuracy: 0.9519612929321352
Test Accuracy: 0.9516381478289925
              precision    recall  f1-score   support

           0       0.92      0.99      0.95     57095
           1       0.99      0.92      0.95     56631

    accuracy                           0.95    113726
   macro avg       0.95      0.95      0.95    113726
weighted avg       0.95      0.95      0.95    113726
```



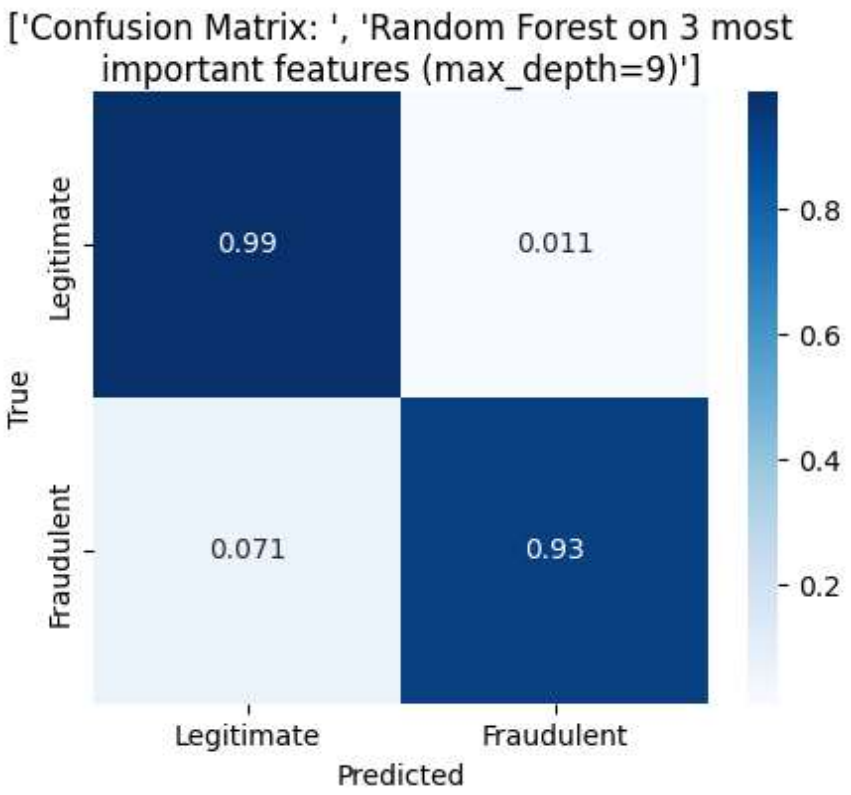

```
Random Forest on 3 most important features (max_depth=7)
Train Accuracy: 0.9553752000422067
Test Accuracy: 0.9558060601797302
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     56605
           1       0.99      0.92      0.95     57121

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```



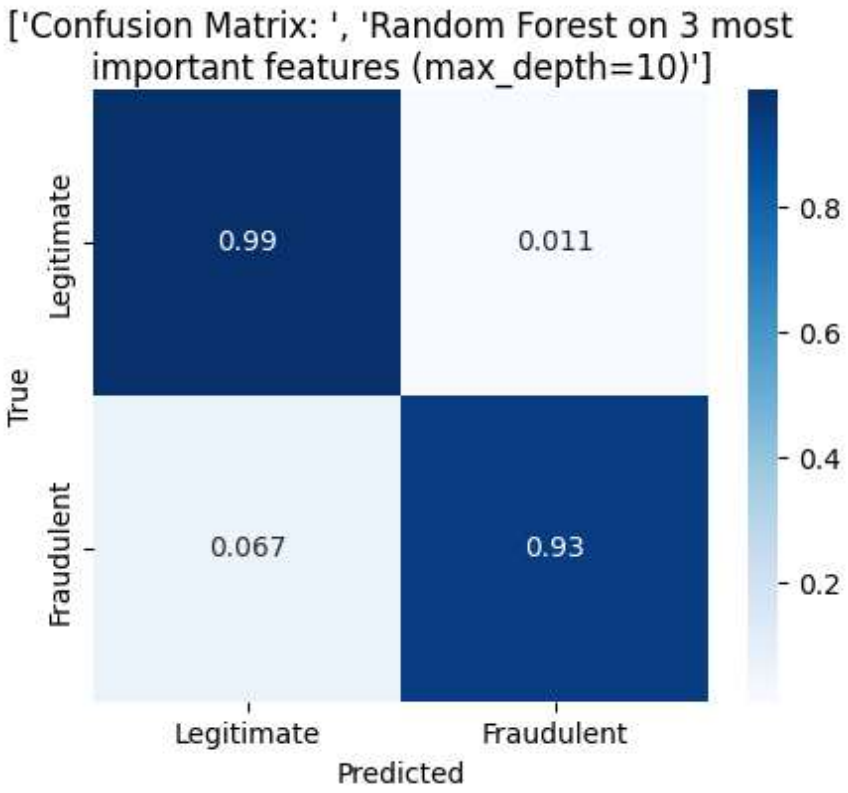

```
Random Forest on 3 most important features (max_depth=8)
Train Accuracy: 0.9584659620491356
Test Accuracy: 0.958268118108436
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     56943
           1       0.99      0.93      0.96     56783

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```




```
Random Forest on 3 most important features (max_depth=9)
Train Accuracy: 0.9599783690624835
Test Accuracy: 0.9590067354870478
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     57059
           1       0.99      0.93      0.96     56667

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```




```
Random Forest on 3 most important features (max_depth=10)
Train Accuracy: 0.9625877108137102
Test Accuracy: 0.9608444858695461
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     57132
           1       0.99      0.93      0.96     56594

    accuracy                           0.96    113726
   macro avg       0.96      0.96      0.96    113726
weighted avg       0.96      0.96      0.96    113726
```



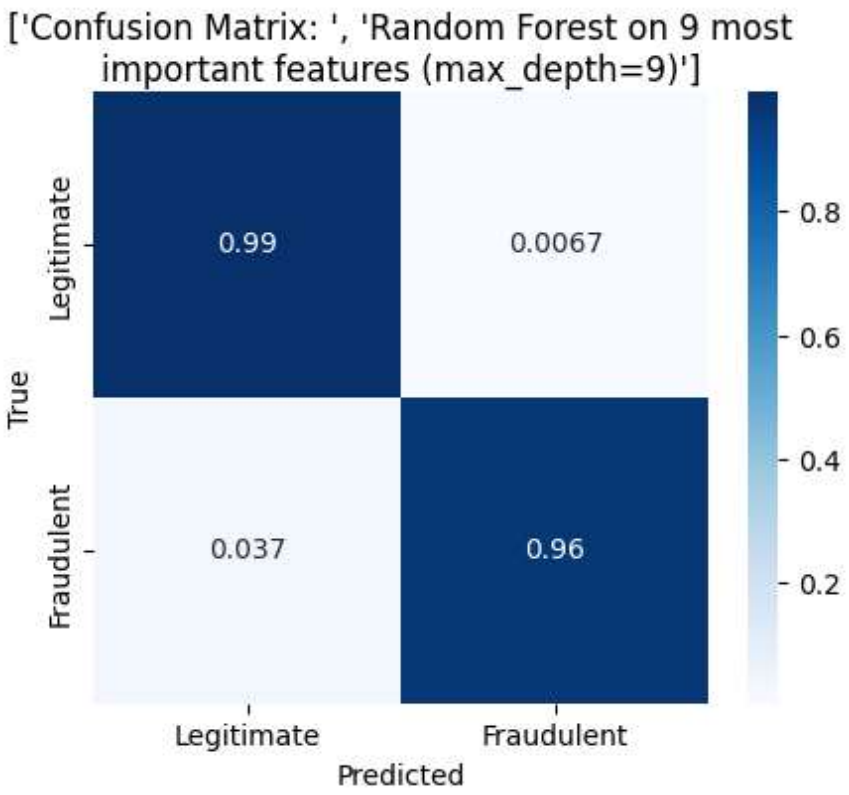
Best Model

Random Forest on 9 most important features with either a max tree depth of 9 or 10 is the best fit for the data.

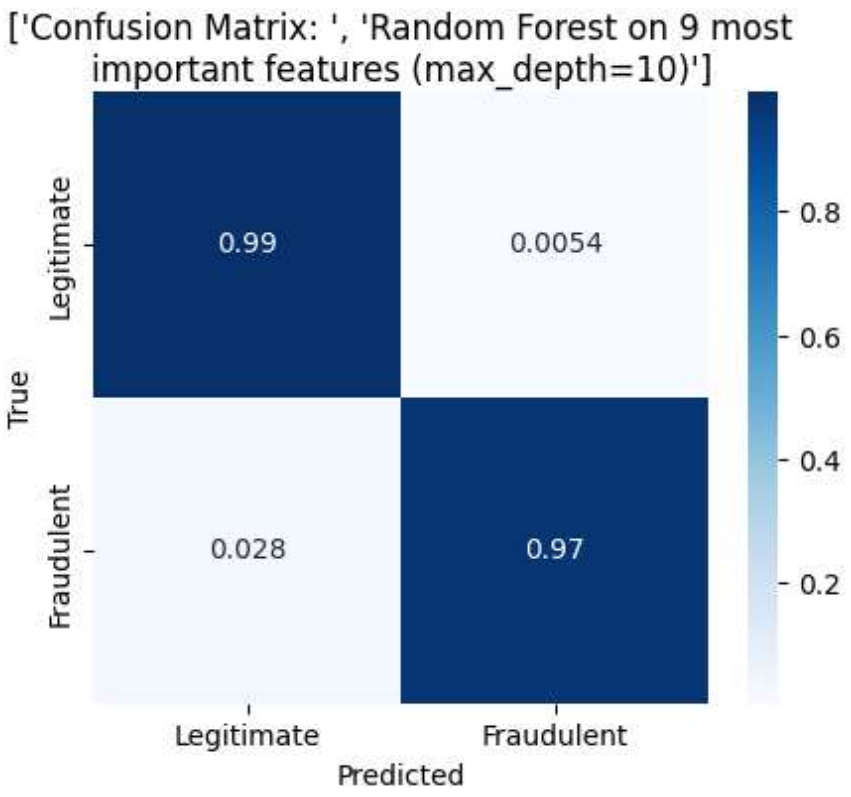
having 100% true negative rate could be suspicious?

In [ ]:
```
important_features_9 = df[['V4', 'V10', 'V12', 'V14', 'V17', 'V11', 'V3', 'V16', 'V2

rf_9_imp_9d, train_acc_9_imp_9d, test_acc_9_imp_9d, X_test_acc_9_imp_9d = train_rand
rf_9_imp_10d, train_acc_9_imp_10d, test_acc_9_imp_10d, X_test_acc_9_imp_10d = train_
```

```
Random Forest on 9 most important features (max_depth=9)
Train Accuracy: 0.9789735856356506
Test Accuracy: 0.9781228566906425
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     56896
           1       0.99      0.96      0.98     56830

    accuracy                           0.98    113726
   macro avg       0.98      0.98      0.98    113726
weighted avg       0.98      0.98      0.98    113726
```




```
Random Forest on 9 most important features (max_depth=10)
Train Accuracy: 0.983948261611241
Test Accuracy: 0.9832843852768935
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     56899
           1       0.99      0.97      0.98     56827

    accuracy                           0.98    113726
   macro avg       0.98      0.98      0.98    113726
weighted avg       0.98      0.98      0.98    113726
```


['Confusion Matrix: ', 'Random Forest on 9 most important features (max_depth=10)']
Legitimate
Fraudulent
True
0.99
0.0054
0.028
0.97
0.8
0.6
0.4
0.2
Legitimate
Fraudulent
Predicted