In [ ]:
```
# core
import pandas as pd
import numpy as np
import joblib
# reading data
import os
from zipfile import ZipFile
from kaggle.api.kaggle_api_extended import KaggleApi
# visualization
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
# data processing & modeling
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
# custom
from scripts.utils import config, evaluate
```

In [ ]:
```
# Kaggle authentication
api = KaggleApi()
api.authenticate()
```

EDA

In [ ]:
```
# download dataset with Kaggle API
api.dataset_download_file('rupakroy/online-payments-fraud-detection-dataset/','PS_2

# designate downloaded file as zip, and unzip
zf = ZipFile('PS_20174392719_1491204439457_log.csv.zip')
zf.extractall()
zf.close()

# read in extracted csv as pandas df
fraud = pd.read_csv('PS_20174392719_1491204439457_log.csv')

# delete downloaded zip and extracted csv - keep your directory clean!
os.remove('PS_20174392719_1491204439457_log.csv.zip')
os.remove('PS_20174392719_1491204439457_log.csv')

print(fraud.shape)

fraud.head()
```

```
(6362620, 11)
```

Out[ ]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703

Column Descriptions

The below column reference:

1. step: represents a unit of time where 1 step equals 1 hour
2. type: type of online transaction
3. amount: the amount of the transaction
4. nameOrig: customer starting the transaction
5. oldbalanceOrg: balance before the transaction
6. newbalanceOrig: balance after the transaction
7. nameDest: recipient of the transaction
8. oldbalanceDest: initial balance of recipient before the transaction
9. newbalanceDest: the new balance of recipient after the transaction
10. isFraud: fraud transaction

In [ ]:
```
for col in fraud.columns:
    print(f'{col} - Dtype: {fraud[col].dtype} - Unique: {fraud[col].nunique()}')
```

```
step - Dtype: int64 - Unique: 743
type - Dtype: object - Unique: 5
amount - Dtype: float64 - Unique: 5316900
nameOrig - Dtype: object - Unique: 6353307
oldbalanceOrg - Dtype: float64 - Unique: 1845844
newbalanceOrig - Dtype: float64 - Unique: 2682586
nameDest - Dtype: object - Unique: 2722362
oldbalanceDest - Dtype: float64 - Unique: 3614697
newbalanceDest - Dtype: float64 - Unique: 3555499
isFraud - Dtype: int64 - Unique: 2
isFlaggedFraud - Dtype: int64 - Unique: 2
```

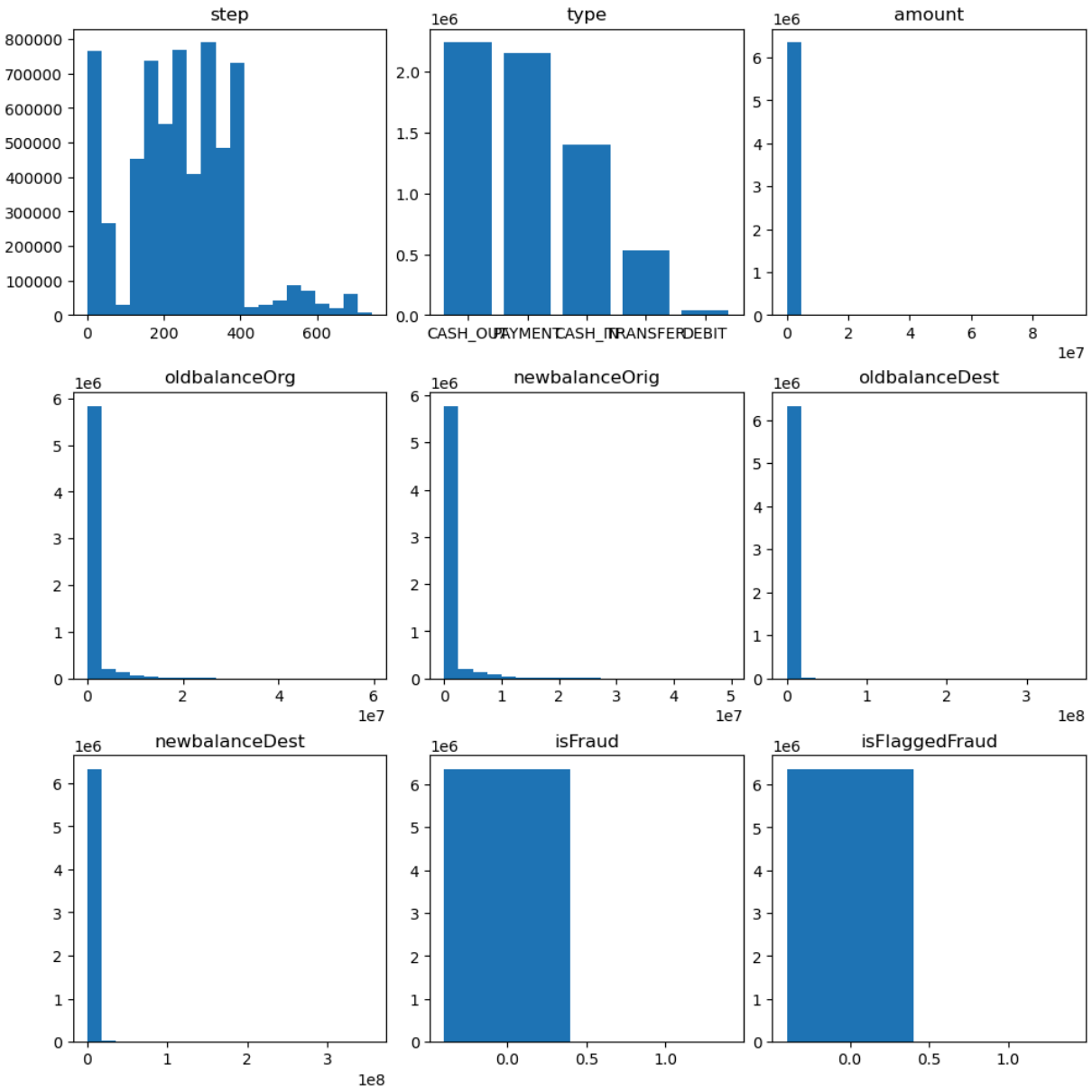
In [ ]:
```
numerical_features = ['step', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbala
categorical_features = ['type', 'nameOrig', 'nameDest', 'isFraud', 'isFlaggedFraud'

fig, axes = plt.subplots(3, 3, figsize=(10, 10), layout='constrained')
cols_to_plot = [x for x in fraud.columns if x != 'nameOrig' and x != 'nameDest']

for col, ax in zip(cols_to_plot, axes.ravel()):
    if col in numerical_features:
        ax.hist(fraud[col], bins=20)
```

```
        ax.set_title(col)
    else:
        ax.bar(fraud[col].value_counts().index, fraud[col].value_counts().values)
        ax.set_title(col)

plt.show()
```



In [ ]:
```
f'Percentage of fraudulent transactions: {fraud[fraud["isFraud"] == 1].shape[0] / f
```

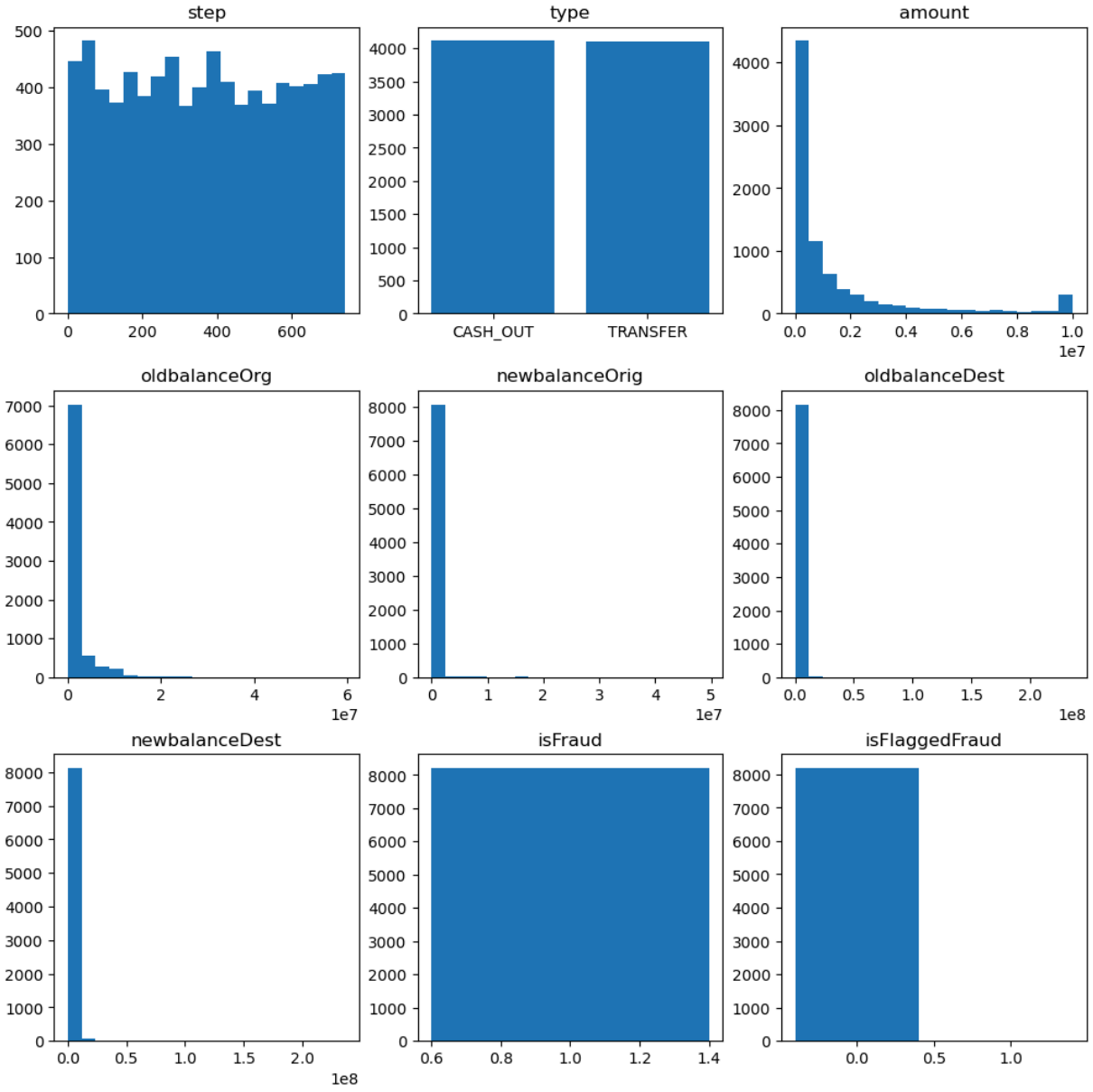
Out[ ]:
```
'Percentage of fraudulent transactions: 0.13%'
```

In [ ]:
```
fig, axes = plt.subplots(3, 3, figsize=(10, 10), layout='constrained')
fraud_filterd = fraud[fraud['isFraud'] == 1]

for col, ax in zip(cols_to_plot, axes.ravel()):
    if col in numerical_features:
        ax.hist(fraud_filterd[col], bins=20)
        ax.set_title(col)
    else:
```

```
        ax.bar(fraud_filterd[col].value_counts().index, fraud_filterd[col].value_co
        ax.set_title(col)

plt.show()
```



Preprocessing

In [ ]:
```
%run scripts/preprocess.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.8
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>
Initial shape: (6362620, 11)

-- Train, Valid, Test Split --

Features
Train: (5090096, 9) - Valid: (636262, 9) - Test (636262, 9)

Labels
Train: (5090096,) - Valid: (636262,) - Test (636262,)

Time Elapsed: 0.23 min


-- Scaling / Encoding --

Features to encode: ['type']
Features to scale: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']

Train: (5090096, 10) - Valid: (636262, 10) - Test (636262, 10)

Time Elapsed: 0.25 min

Preprocessing Complete. Time Elapsed: 0.26 min
```

Baseline Training

In [ ]:
```
%run scripts/train.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.8
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>

-- Fitting RandomForestClassifier --

Performance on Training Subset
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1: 1.00

Performance on Validation Subset
Accuracy: 1.00
Precision: 0.94
Recall: 0.90
F1: 0.92

Time Elapsed: 1.48 min


-- Fitting SVC --

Performance on Training Subset
Accuracy: 1.00
Precision: 0.99
Recall: 0.46
F1: 0.63

Performance on Validation Subset
Accuracy: 1.00
Precision: 1.00
Recall: 0.46
F1: 0.63

Time Elapsed: 162.50 min

Training Complete - 162.50 min
```

With small subset of training data (for time)

In [ ]:
```
# Edited utils.py to set 'TRAIN_SIZE' equal to 0.01, re-process and re-train
%run scripts/preprocess.py
%run scripts/train.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.05
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>
Initial shape: (6362620, 11)

-- Train, Valid, Test Split --

Features
Train: (318131, 9) - Valid: (3022244, 9) - Test (3022245, 9)

Labels
Train: (318131,) - Valid: (3022244,) - Test (3022245,)

Time Elapsed: 0.26 min


-- Scaling / Encoding --

Features to encode: ['type']
Features to scale: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']

Train: (318131, 10) - Valid: (3022244, 10) - Test (3022245, 10)

Time Elapsed: 0.28 min

Preprocessing Complete. Time Elapsed: 0.29 min

-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.05
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>

-- Fitting RandomForestClassifier --

Performance on Training Subset
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1: 1.00
```

```
Performance on Validation Subset
Accuracy: 1.00
Precision: 0.92
Recall: 0.77
F1: 0.84

Time Elapsed: 0.10 min


-- Fitting SVC --

Performance on Training Subset
Accuracy: 1.00
Precision: 0.99
Recall: 0.34
F1: 0.51

Performance on Validation Subset
Accuracy: 1.00
Precision: 1.00
Recall: 0.32
F1: 0.48

Time Elapsed: 4.79 min

Training Complete - 4.79 min
```

Hyperparameter Tuning

Due to long training times, I skipped a full grid search, instead looking at results when changing key hyperparameters individually. This is unlikely to drive optimal results, but is a necessary compromise to keep training times reasonable.

For each call of the tuning.py script below, I've edited the space of hyperparameters in utils.py in line with the comments in each cell.

The below dictionary summarizes the hyperparameter space.

```
    RANDOM_SEED = 42

    hyperparams = {
        'svm': {
            'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
            'C': [0.1, 1, 10, 100],
            'gamma': ['scale', 'auto', 0.1, 1, 10],
            'ceof0': [0, 0.5, 1],
            'random_state': [RANDOM_SEED],
        }
    }
```

In [ ]:
```
# 'kernel': ['linear', 'poly', 'rbf', 'sigmoid']
%run scripts/tuning.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.05
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>

--Total number of hyperparameter combinations for svm: 4
Completed 1 of 4 hyperparameter combinations Time in Loop: 0.59 min
Completed 2 of 4 hyperparameter combinations Time in Loop: 1.79 min
Completed 3 of 4 hyperparameter combinations Time in Loop: 6.50 min
Completed 4 of 4 hyperparameter combinations Time in Loop: 7.62 min
SVM tuning complete - 7.62 min
Tuning complete - 7.62 min
```

In [ ]:
```
svm_results = pd.read_excel('logs/tuning_results_kernel.xlsx', engine='openpyxl', i
display(svm_results.sort_values(by='valid_f1', ascending=False).T)
```

	2	0	1	3
model	svm	svm	svm	svm
train_accuracy	0.999145	0.999101	0.999082	0.998098
train_precision	0.992908	0.977099	0.991736	0.019802
train_recall	0.340633	0.311436	0.291971	0.009732
train_f1	0.507246	0.472325	0.451128	0.013051
valid_accuracy	0.999119	0.999075	0.999069	0.998082
valid_precision	0.995994	0.983392	0.972222	0.019747
valid_recall	0.318636	0.288388	0.287106	0.009997
valid_f1	0.482812	0.445986	0.443301	0.013274
kernel	rbf	linear	poly	sigmoid
random_state	42	42	42	42

RBF shows the best overall performance.

In [ ]:
```
# 'kernel': ['rbf']
# 'C': [0.1, 1, 10, 100]
%run scripts/tuning.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.05
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>

--Total number of hyperparameter combinations for svm: 4
Completed 1 of 4 hyperparameter combinations Time in Loop: 10.58 min
Completed 2 of 4 hyperparameter combinations Time in Loop: 15.34 min
Completed 3 of 4 hyperparameter combinations Time in Loop: 17.67 min
Completed 4 of 4 hyperparameter combinations Time in Loop: 20.13 min
SVM tuning complete - 20.13 min
Tuning complete - 20.13 min
```

In [ ]:
```
svm_results = pd.read_excel('logs/tuning_results_C.xlsx', engine='openpyxl', index_
display(svm_results.sort_values(by='valid_f1', ascending=False).T)
```

	3	2	1	0
model	svm	svm	svm	svm
train_accuracy	0.99944	0.999299	0.999145	0.998906
train_precision	0.979424	0.994737	0.992908	1.0
train_recall	0.579075	0.459854	0.340633	0.153285
train_f1	0.727829	0.628952	0.507246	0.265823
valid_accuracy	0.999401	0.999263	0.999119	0.998865
valid_precision	0.97849	0.991192	0.995994	1.0
valid_recall	0.548065	0.43271	0.318636	0.120482
valid_f1	0.702596	0.602427	0.482812	0.215054
C	100.0	10.0	1.0	0.1
kernel	rbf	rbf	rbf	rbf
random_state	42	42	42	42

In [ ]:
```
# 'kernel': ['rbf']
# 'C': [100]
# 'gamma': ['scale', 'auto', 0.1, 1, 10],
%run scripts/tuning.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.05
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>

--Total number of hyperparameter combinations for svm: 5
Completed 1 of 5 hyperparameter combinations Time in Loop: 2.44 min
Completed 2 of 5 hyperparameter combinations Time in Loop: 4.26 min
Completed 3 of 5 hyperparameter combinations Time in Loop: 6.07 min
Completed 4 of 5 hyperparameter combinations Time in Loop: 11.09 min
Completed 5 of 5 hyperparameter combinations Time in Loop: 46.36 min
SVM tuning complete - 46.36 min
Tuning complete - 46.36 min
```

In [ ]:
```
svm_results = pd.read_excel('logs/tuning_results_gamma.xlsx', engine='openpyxl', in
display(svm_results.sort_values(by='valid_f1', ascending=False).T)
```

	3	4	0	1	2
model	svm	svm	svm	svm	svm
train_accuracy	0.999544	0.999601	0.99944	0.999368	0.999368
train_precision	0.985401	0.986301	0.979424	0.981651	0.981651
train_recall	0.656934	0.70073	0.579075	0.520681	0.520681
train_f1	0.788321	0.819346	0.727829	0.680445	0.680445
valid_accuracy	0.999468	0.999457	0.999401	0.99935	0.99935
valid_precision	0.973174	0.96348	0.97849	0.984492	0.984492
valid_recall	0.60446	0.601897	0.548065	0.504486	0.504486
valid_f1	0.745731	0.740928	0.702596	0.667119	0.667119
C	100	100	100	100	100
gamma	1	10	scale	auto	0.1
kernel	rbf	rbf	rbf	rbf	rbf
random_state	42	42	42	42	42

In [ ]:
```
# 'kernel': ['rbf']
# 'C': [100]
# 'gamma': [1],
# 'coef0': [0, 0.5, 1],
%run scripts/tuning.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.05
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>

--Total number of hyperparameter combinations for svm: 3
Completed 1 of 3 hyperparameter combinations Time in Loop: 4.97 min
Completed 2 of 3 hyperparameter combinations Time in Loop: 9.93 min
Completed 3 of 3 hyperparameter combinations Time in Loop: 14.88 min
SVM tuning complete - 14.88 min
Tuning complete - 14.88 min
```

In [ ]:
```
svm_results = pd.read_excel('logs/tuning_results_coef0.xlsx', engine='openpyxl', in
display(svm_results.sort_values(by='valid_f1', ascending=False).T)
```

	0	1	2
model	svm	svm	svm
train_accuracy	0.999544	0.999544	0.999544
train_precision	0.985401	0.985401	0.985401
train_recall	0.656934	0.656934	0.656934
train_f1	0.788321	0.788321	0.788321
valid_accuracy	0.999468	0.999468	0.999468
valid_precision	0.973174	0.973174	0.973174
valid_recall	0.60446	0.60446	0.60446
valid_f1	0.745731	0.745731	0.745731
C	100	100	100
coef0	0.0	0.5	1.0
gamma	1	1	1
kernel	rbf	rbf	rbf
random_state	42	42	42

Final Run with Full Data and Optimal Hyperparameters

In [ ]:
```
# Edit utils.py to reset 'TRAIN_SIZE' equal to 0.9, re-process and re-train
%run scripts/preprocess.py
```

```
-- Config --

RANDOM_SEED: 42
TRAIN_SIZE: 0.9
SHUFFLE: True
STRATIFY: True
FEATURES_TO_REMOVE: ['nameOrig', 'nameDest', 'isFlaggedFraud']
FEATURES_TO_SCALE: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']
FEATURES_TO_ENCODE: ['type']
SCALER: <class 'sklearn.preprocessing._data.StandardScaler'>
Initial shape: (6362620, 11)

-- Train, Valid, Test Split --

Features
Train: (5726358, 9) - Valid: (318131, 9) - Test (318131, 9)

Labels
Train: (5726358,) - Valid: (318131,) - Test (318131,)

Time Elapsed: 0.15 min


-- Scaling / Encoding --

Features to encode: ['type']
Features to scale: ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']

Train: (5726358, 10) - Valid: (318131, 10) - Test (318131, 10)

Time Elapsed: 0.17 min

Preprocessing Complete. Time Elapsed: 0.17 min
```

In [ ]:
```
hyperparams = {
    'rf': {
        'n_estimators': 100,
        'criterion': 'entropy',
        'max_depth': None,
        'max_features': 1.0,
        'random_state': config['RANDOM_SEED'],
        'n_jobs': -1
    },
    'svm': {
        'kernel': 'rbf',
        'C': 100,
        'gamma': 1,
        'degree': 0,
        'coef0': 0,
        'random_state': config['RANDOM_SEED'],
    }
```

```
}

X_train = np.load('data/processed/X_train.npy')
y_train = np.load('data/processed/y_train.npy')

X_valid = np.load('data/processed/X_valid.npy')
y_valid = np.load('data/processed/y_valid.npy')

svm = SVC(**hyperparams['svm'])

models = [svm]

for model in models:
    print(f'\n-- Fitting {model.__class__.__name__} --\n')
    # Train model
    model.fit(X_train, y_train)
    # Get evaluation metrics
    print('Performance on Training Subset')
    train_accuracy, train_precision, train_recall, train_f1 = evaluate(model, X_tra
    print('Performance on Validation Subset')
    valid_accuracy, valid_precision, valid_recall, valid_f1 = evaluate(model, X_val
    # Log results
    result = pd.DataFrame({
        'model': model.__class__.__name__,
        'train_accuracy': train_accuracy,
        'train_precision': train_precision,
        'train_recall': train_recall,
        'train_f1': train_f1,
        'valid_accuracy': valid_accuracy,
        'valid_precision': valid_precision,
        'valid_recall': valid_recall,
        'valid_f1': valid_f1
    }, index=[0])
```

```
-- Fitting SVC --

Performance on Training Subset
Accuracy: 1.00
Precision: 0.98
Recall: 0.69
F1: 0.81

Performance on Validation Subset
Accuracy: 1.00
Precision: 0.98
Recall: 0.66
F1: 0.79
```

In [ ]:

```
X_train = np.load('data/processed/X_train.npy')
y_train = np.load('data/processed/y_train.npy')

X_valid = np.load('data/processed/X_valid.npy')
y_valid = np.load('data/processed/y_valid.npy')

X_train = np.concatenate([X_train, X_valid])
```

```
y_train = np.concatenate([y_train, y_valid])

X_test = np.load('data/processed/X_test.npy')
y_test = np.load('data/processed/y_test.npy')

print(
    f'Final training set size: {X_train.shape} - {y_train.shape}'
    f'\nFinal test set size: {X_test.shape} - {y_test.shape}'
)
```

```
Final training set size: (6044489, 10) - (6044489,)
Final test set size: (318131, 10) - (318131,)
```

In [ ]:

```
hyperparams = {
    'rf': {
        'n_estimators': 100,
        'criterion': 'entropy',
        'max_depth': None,
        'max_features': 1.0,
        'random_state': config['RANDOM_SEED'],
        'n_jobs': -1
    },
    'svm': {
        'kernel': 'rbf',
        'C': 100,
        'gamma': 1,
        'degree': 0,
        'coef0': 0,
        'random_state': config['RANDOM_SEED'],
    }
}

rf = RandomForestClassifier(**hyperparams['rf'])
svm = SVC(**hyperparams['svm'])

models = [rf, svm]

for model in models:
    print(f'\n-- Fitting {model.__class__.__name__} --\n')
    # Train model
    model.fit(X_train, y_train)
    # Get evaluation metrics
    print('Performance on Training + Validation Subset')
    train_accuracy, train_precision, train_recall, train_f1 = evaluate(model, X_tra
    print('Performance on Test Subset')
    test_accuracy, test_precision, test_recall, test_f1 = evaluate(model, X_test, y
    # Log results
    result = pd.DataFrame({
        'model': model.__class__.__name__,
        'train_accuracy': train_accuracy,
        'train_precision': train_precision,
        'train_recall': train_recall,
        'train_f1': train_f1,
        'test_accuracy': test_accuracy,
        'test_precision': test_precision,
        'test_recall': test_recall,
```

```
        'test_f1': test_f1
    }, index=[0])
```

```
-- Fitting RandomForestClassifier --

Performance on Training + Validation Subset
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1: 1.00

Performance on Test Subset
Accuracy: 1.00
Precision: 0.96
Recall: 0.90
F1: 0.93


-- Fitting SVC --

Performance on Training + Validation Subset
Accuracy: 1.00
Precision: 0.98
Recall: 0.69
F1: 0.81

Performance on Test Subset
Accuracy: 1.00
Precision: 0.98
Recall: 0.69
F1: 0.81
```

In [ ]:
```
joblib.dump(rf, 'models/rf.joblib')
joblib.dump(svm, 'models/svm.joblib')
```

Out[ ]:
```
['models/svm.joblib']
```