Predicting Tipping Behavior using NYC Green Taxi trip Data

APPLICATIONS OF BINARY CLASSIFICATION MODELS

WEINING HU

APRIL 26, 2020

Table of Content

Problem Statement

Background and Predictive Task

In 2018, almost a million taxi trips were made daily in the city of New York, producing a plethora of information that proves useful for drivers. **Predicting tipping behavior** is particularly useful as it can help taxi drivers **understand factors driving tipping actions**, and therefore can improve their business and maximize the tip received.

Literature Review

Han, Jiamin. "**Looking Through the Taxi Meter - Analysis of the NYC Green Taxi Data**." 2019

- Detailed sample selection process
- Predicted tip percentage using OLS linear regression model from the *Statsmodels* package
- Prediction performance was unsatisfactory due to structural issues within the dataset

Kthouz. "**NYC Green Taxi**." 2017

- Used logistic regression model to classify data into tipper and stiffer (riders who don't tip)
- Improved model performance using Gradient Boosting Classifier
- Limitation: did not remove non-credit card transactions from the raw set

Binary Classification Problem	Credit Card Transactions	Model Ensembles

Dataset Description

- New York City Taxi and Limousine Commission (TLC)
- Green taxi trips in June 2019 (hereafter 'raw dataset')
- 471,052 observations
- 20 columns
  - **Admin related (2)**: VendorID, Store_and_fwd_flag
  - **Trip related (7)**: lpep_pickup_datetime, lpep_dropoff_datetime, passenger_count, Trip_distance, PULocationID, DOLocationID, Trip_type
  - **Trip fare related (11)**: RateCodeID, Payment_type, Fare_amount, Extra, MTA_tax, Improvement_surcharge, Tip_amount, Tolls_amount, Total_amount, congestion surcharge, ehail_fee
- Feature datatypes: categorical, numeric (int64 and float64)

- Taxi Zone Shapefile
- Taxi Zone Lookup Table

locationid	borough	zone	Service zone
1	Manhattan	Hamilton Heights	Boro Zone
2	Queens	Jamaica Bay	Boro Zone

Note:
5 boroughs (Queens, Bronx, Manhattan, Staten Island, and Brooklyn)
263 taxi zones
3 service zones (Boro Zone, Yellow Zone, and Airports.)

Feature Engineering

Location

PULocationID	DOLocationID
74	263
75	74

df.merge

pu boro	pu zone	pu sz
Manhattan	East Harlem North	Boro Zone
Manhattan	East Harlem South	Boro Zone

Time

PU Datetime	DO datetime
2019-06-01 00:25:27	2019-06-01 00:33:52
2019-06-01 00:39:13	2019-06-01 00:46:38

pd.to_datetime
df.dt.hour
df.dt.weekday

pu hour	pu day	trip time
0	5 (Saturday)	8.0 (mins)
0	5 (Saturday)	7.0

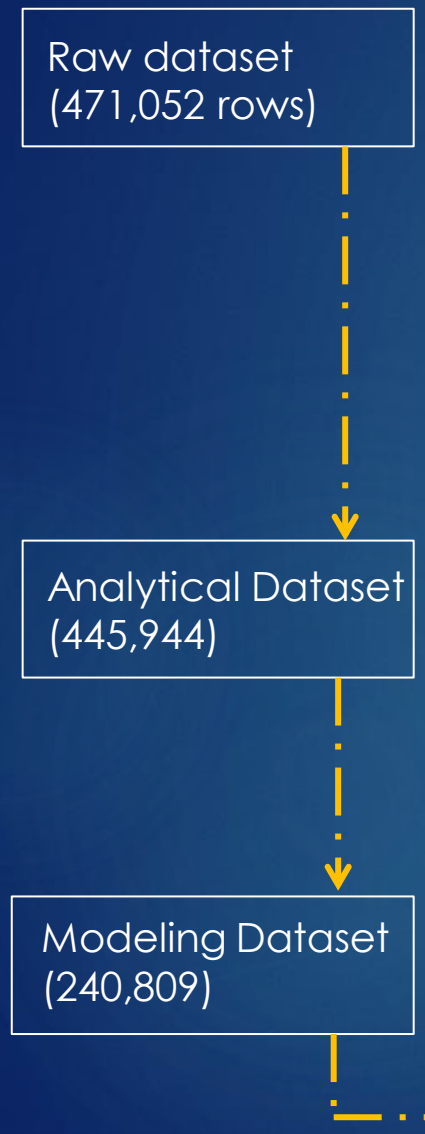
Segment Data into Bins

Passenger Count
1
9

Pd.cut(x, bins, labels)

Passenger_range
'1-2'
'5+'

Data Preprocessing



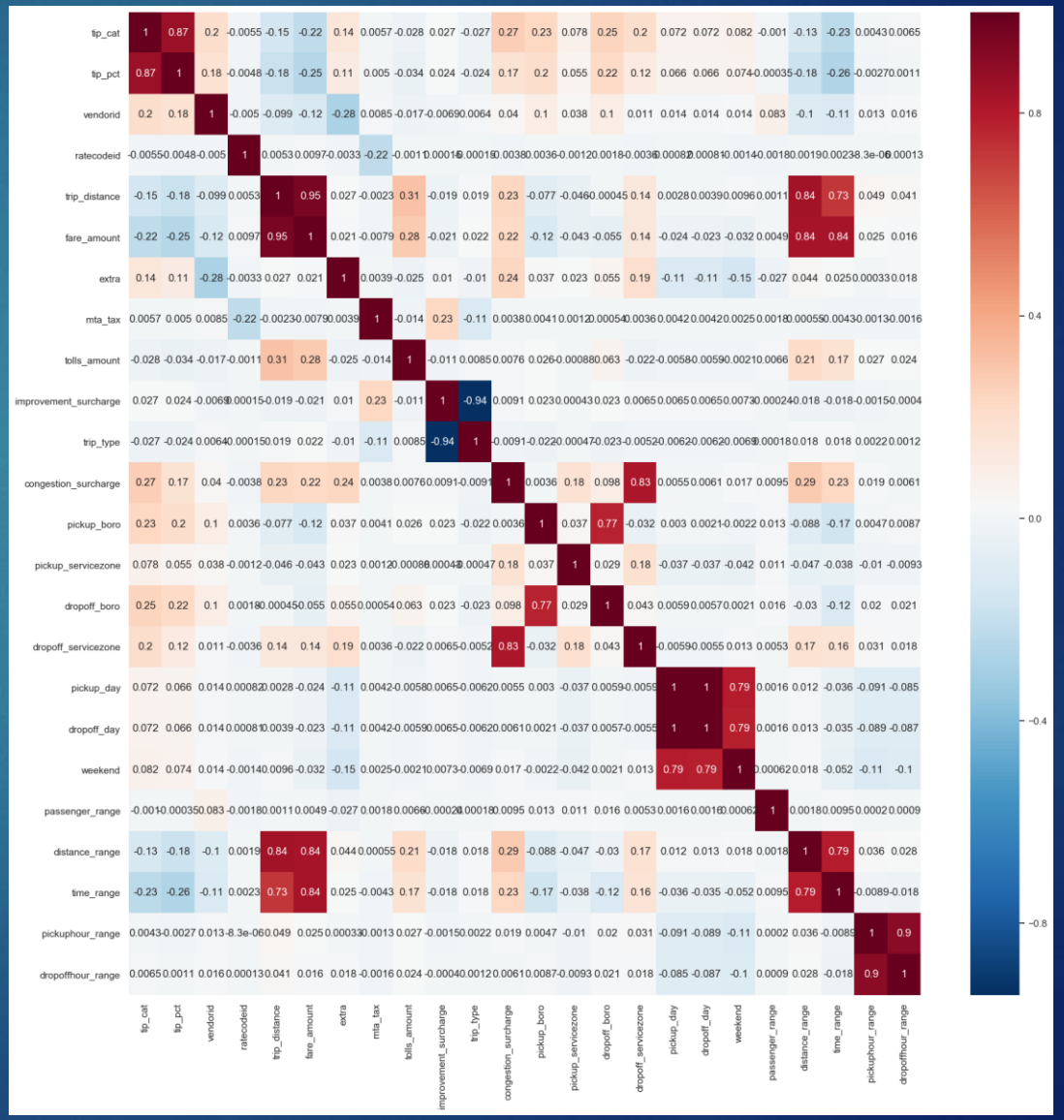
Remove Criteria	# of Rows	Pct of RD
Distance of 0 mile (invalid obs only) or > 17.44 miles (99% pctl)	10,733	2.3%
Trip time of 0 min (invalid obs only) or > 70 minutes (99% pctl)	8,175	1.75%
Fare amount < $2.50 or > $44.5 (99% pctl)	6,105	1.31%
Tip amount > twice the fare amount	76	0.02%
Trip speed >= 240 MPH	19	~0.0001%

Impute Data	# of Rows	Pct of RD
Impute valid observations with distance of 0 mile and/or 0 min	970	0.002%
Replace ['passenger_count'] == 0 by 1 (mode)	642	0.001%

Filter Observations	# of Rows	Pct of RD
Select credit card transaction observations (payment type = 1)	240,809	51.12%

Feature Selection

- Drop repetitive features
  - pickup_day vs dropoff_day
  - pickup_hour vs dropoff_hour
- Drop highly correlated features
  - Trip_speed vs (trip_time & trip distance)
- Drop irrelevant features
  - i.e. ehail_fee, payment_type



Binary Classification Models

- Model Selection Process
  - Encode target labels {0: non-tipper, 1: tipper}
  - Training set (70%), testing set (30%)
  - StratifiedKfold = 3
  - GridSearchCV(estimator, parameter_grid)
    - Classifiers: Decision Tree, Random Forest, AdaBoost, XGBoost, SVM
  - Validation Evaluation Matrix = f1-score
  - Processing Time = ~6 hrs
    - using an i5 laptop with 8G Mem

Rank	Best Score (F1-Score)	Classifier
1	0.8911	XGBoost
2	0.8870	AdaBoost
3	0.8736	Random Forest
4	0.8619	SVM
5	0.8501	Decision Tree

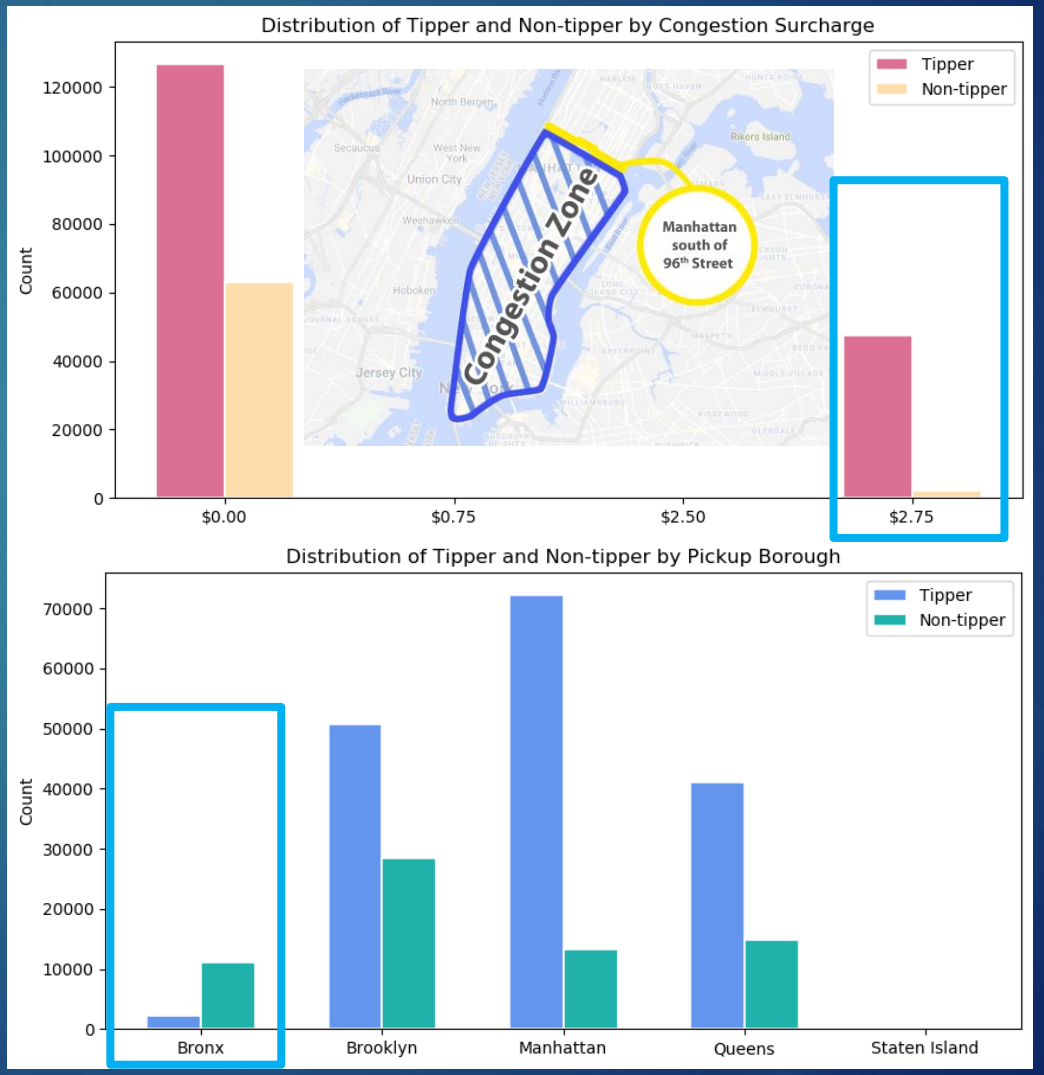
Analysis: Evaluation and Test Results

Validation Results			Test Results (Weighted)			
Rank	Classifier	F1-Score	Precision	Recall	F1-Score	Accuracy
1	XGBoost	0.89	0.83	0.83	0.82	0.83
2	AdaBoost	0.89	0.82	0.83	0.82	0.83
3	Random Forest	0.87	0.82	0.81	0.82	0.81
4	SVM	0.86	0.81	0.80	0.80	0.80
5	Decision Tree	0.85	0.81	0.80	0.80	0.80

Model Boosting

Ensembled Model (All Classifiers)	**0.84**	**0.84**	**0.83**	**0.84**

Analysis: Factors Driving Tipping Behaviors



Congestion Surcharge Chart

Conclusion

Factors driving passengers tipping preference

- Congestion surcharge (More tips in congested area)
- Pickup location (Less tips in the Bronx)
- Trip duration (More tips in shorter duration)

Target the following passengers (suggestions for drivers)

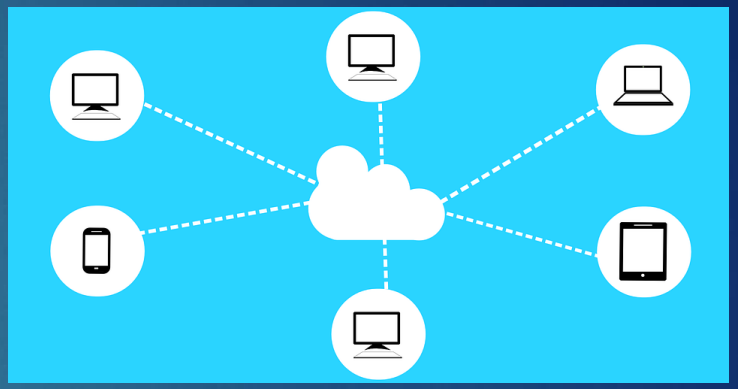
Lower Manhattan!

Avoid Bronx!

Shorter Duration!

Limitation and Feature Research

- Limitation
  - Limited computational resources
  - Reflect only seasonal patterns



- Future work
  - Integrate more Socio-demographic information
    - Income level of the taxi service zone
    - Past tipping rate of passenger
    - Driver's driving style
    - Car condition

Reference

Elliott, David & Tomasini, Marcello & Oliveira, Marcos & Menezes, Ronaldo. (2017). Tippers and Stiffers: an Analysis of Tipping Behavior in Taxi Trips. 10.1109/UIC-ATC.2017.8397523

Gshahane. "NYC Green Taxi Data Visualization." *GitHub*, 16 Feb. 2017, github.com/gshahane/NYC-Green-Taxi-Data-Visualization.

Guo, Yurong, et al. "Predict New York City Taxi Demand." *NYC Data Science Academy*, 21 Sept. 2016, nycdatascience.com/blog/student-works/predict-new-york-city-taxi-demand/.

Han, Jiamin. "Looking Through the Taxi Meter - Analysis of the NYC Green Taxi Data." *Medium*, Medium, 4 Mar. 2019, medium.com/@jiaminhan/looking-through-the-taxi-meter-analysis-of-the-nyc-green-taxi-data-c1dbe5619afe.

Kthouz. "NYC Green Taxi." *GitHub*, 17 Sept. 2017, github.com/kthouz/NYC_Green_Taxi.

Miles, Christian. "Visualizing NYC Taxi Cab Data." *Cambridge Intelligence*, 7 Apr. 2020, cambridge-intelligence.com/visualizing-nyc-taxi-cab-data/.

R-Shekhar. "r-Shekhar/NYC-Transport." *GitHub*, 18 June 2017, github.com/r-shekhar/NYC-transport/blob/master/15_dataframe_analysis/spatialjoin_geopandas_dask.ipynb.

Appendix

	precision	recall	f1-score	support
0	0.80	0.50	0.62	19334
1	0.84	0.95	0.89	52566
accuracy			0.83	71900
macro avg	0.82	0.73	0.75	71900
weighted avg	0.83	0.83	0.82	71900

```
[[ 9752  9582]
 [ 2500 50066]]
```

Figure 1. XGBoost Classifier

	precision	recall	f1-score	support
0	0.78	0.50	0.61	19334
1	0.84	0.95	0.89	52566
accuracy			0.83	71900
macro avg	0.81	0.73	0.75	71900
weighted avg	0.82	0.83	0.82	71900

```
[[ 9703  9631]
 [ 2688 49878]]
```

Figure 2. Adaboost Classifier

	precision	recall	f1-score	support
0	0.64	0.70	0.67	19334
1	0.89	0.85	0.87	52566
accuracy			0.81	71900
macro avg	0.76	0.78	0.77	71900
weighted avg	0.82	0.81	0.82	71900

```
[[13576  5758]
 [ 7644 44922]]
```

Figure 3. Random Forest Classifier

	precision	recall	f1-score	support
0	0.60	0.72	0.65	19334
1	0.89	0.82	0.85	52566
accuracy			0.80	71900
macro avg	0.74	0.77	0.75	71900
weighted avg	0.81	0.80	0.80	71900

```
[[13938  5396]
 [ 9306 43260]]
```

Figure 4. SVM

	precision	recall	f1-score	support
0	0.61	0.71	0.66	19334
1	0.89	0.83	0.86	52566
accuracy			0.80	71900
macro avg	0.75	0.77	0.76	71900
weighted avg	0.81	0.80	0.80	71900

```
[[13812  5522]
 [ 8892 43674]]
```

Figure 5. Decision Tree Classifier

	precision	recall	f1-score	support
0	0.79	0.56	0.65	19334
1	0.85	0.94	0.90	52566
accuracy			0.84	71900
macro avg	0.82	0.75	0.78	71900
weighted avg	0.84	0.84	0.83	71900

```
[[10805  8529]
 [ 2918 49648]]
```

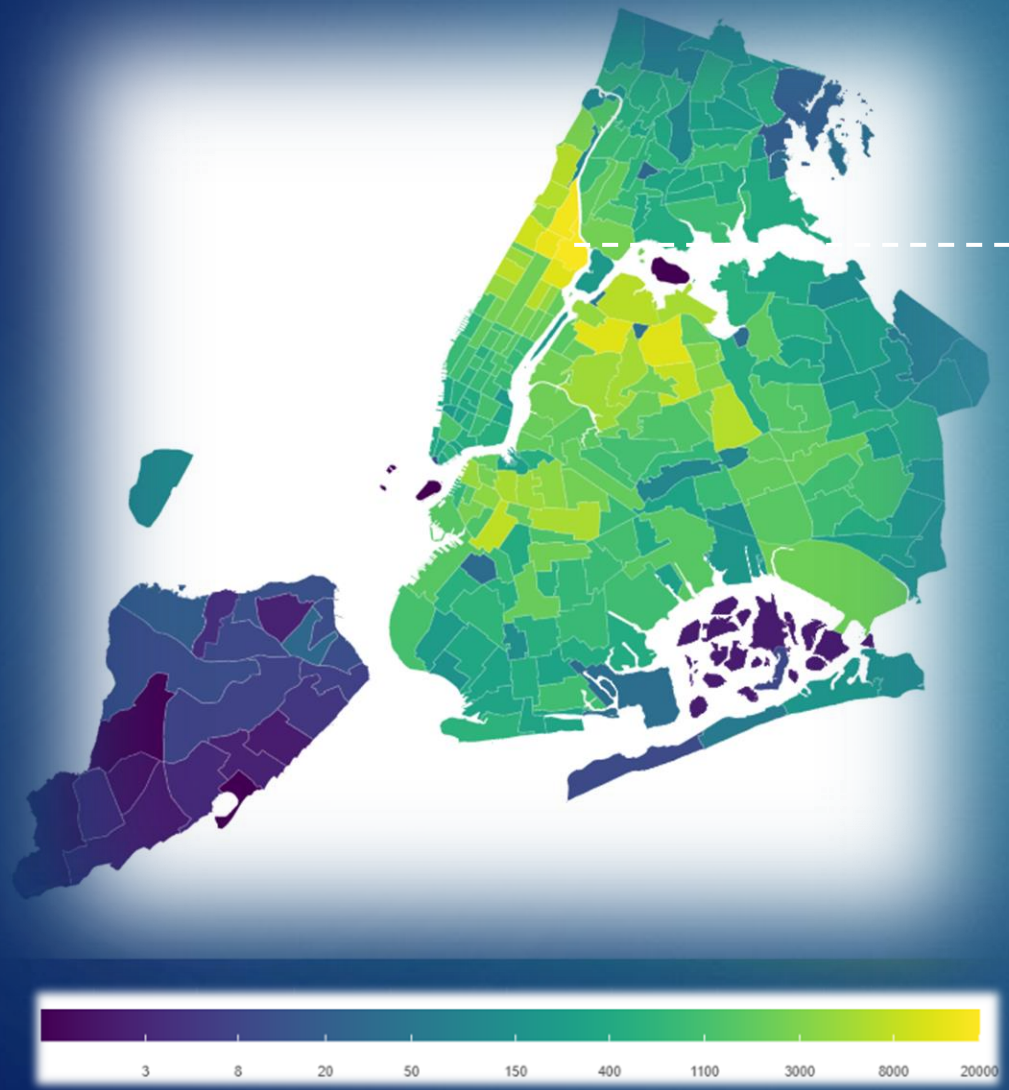
Figure 6. Ensembled Model

Visualization – Popular PU Locations

Name: East Harlem North
Borough: Manhattan
Zone ID: 74
# of Trips: 38,459

Rank	Popular PULoc	# Trips
1	East Harlem North	38,459
2	East Harlem South	29,943
3	Central Harlem	27,901
4	Elmhurst	22,084
5	Astoria	20,849

Visualization – Popular DO Locations



Name: East Harlem North
Borough: Manhattan
Zone ID: 74
# of Trips: 18,586

Rank	Popular DOLoc	# Trips
1	East Harlem North	18,586
2	Central Harlem North	16,854
3	Central Harlem	15,057
4	East Harlem South	12,888
5	Astoria	12,662

```
best_score,best_param,best_estimator
0.8914295843857635,"{'model__eta': 1e-08, 'model__gamma': 0.001, 'model__lambda':
1e-08}","Pipeline(memory=None,
         steps=[('StandardScaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('model',
                 XGBClassifier(base_score=0.5, booster='gbtree',
                               colsample_bylevel=1, colsample_bynode=1,
                               colsample_bytree=1, eta=1e-08, gamma=0.001,
                               lambda=1e-08, learning_rate=0.1,
                               max_delta_step=0, max_depth=3,
                               min_child_weight=1, missing=None,
                               n_estimators=100, n_jobs=1, nthread=None,
                               objective='binary:logistic', random_state=0,
                               reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                               seed=42, silent=None, subsample=1,
                               verbosity=1))],
         verbose=False)"
0.8878531433539013,{'model__learning_rate': 1.2},"Pipeline(memory=None,
         steps=[('StandardScaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('model',
                 AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
                                    learning_rate=1.2, n_estimators=50,
                                    random_state=42))],
         verbose=False)"
```

```
0.8675853696574962,"{'model__min_samples_leaf': 5, 'model__min_samples_split':
2}","Pipeline(memory=None,
         steps=[('StandardScaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('model',
                 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                        class_weight='balanced',
                                        criterion='gini', max_depth=None,
                                        max_features='auto',
                                        max_leaf_nodes=None, max_samples=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=5, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=100, n_jobs=None,
                                        oob_score=False, random_state=42,
                                        verbose=0, warm_start=False))],
         verbose=False)"
0.8549453409408821,"{'model__C': 160, 'model__gamma': 0.0001, 'model__kernel':
'rbf'}","Pipeline(memory=None,
         steps=[('StandardScaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('model',
                 SVC(C=160, break_ties=False, cache_size=200,
                     class_weight='balanced', coef0=0.0,
                     decision_function_shape='ovr', degree=3, gamma=0.0001,
                     kernel='rbf', max_iter=-1, probability=False,
                     random_state=42, shrinking=True, tol=0.001,
                     verbose=False))],
         verbose=False)"
```

```
0.8521548146568808,"{'model__min_samples_leaf': 20, 'model__min_samples_split':
200}","Pipeline(memory=None,
         steps=[('StandardScaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('model',
                 DecisionTreeClassifier(ccp_alpha=0.0, class_weight='balanced',
                                        criterion='gini', max_depth=None,
                                        max_features=None, max_leaf_nodes=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=20,
                                        min_samples_split=200,
                                        min_weight_fraction_leaf=0.0,
                                        presort='deprecated', random_state=42,
                                        splitter='best'))],
         verbose=False)"

```