Chapter 2

Prepare Data: Read in the data

Using the Titanic dataset from this Kaggle competition (we are only using the training set).

This dataset contains information about 891 people who were on board the ship when departed on April 15th, 1912. As noted in the description on Kaggle's website, some people aboard the ship were more likely to survive the wreck than others. There were not enough lifeboats for everybody so women, children, and the upper-class were prioritized. Using the information about these 891 passengers, the challenge is to build a model to predict which people would survive based on the following fields:

- **Name** (str) - Name of the passenger
- **Pclass** (int) - Ticket class
- **Sex** (str) - Sex of the passenger
- **Age** (float) - Age in years
- **SibSp** (int) - Number of siblings and spouses aboard
- **Parch** (int) - Number of parents and children aboard
- **Ticket** (str) - Ticket number
- **Fare** (float) - Passenger fare
- **Cabin** (str) - Cabin number
- **Embarked** (str) - Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

Read in Data

In [2]:
```
import pandas as pd

titanic = pd.read_csv('./titanic.csv')
titanic.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Clean continuous variables

Fill missing for Age

In [4]:
```
titanic.isnull().sum()
```

Out[4]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [5]:
```
titanic['Age'].fillna(titanic['Age'].mean(), inplace=True)
titanic.head(10)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	29.699118	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.000000	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.000000	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.000000	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.000000	1	0	237736	30.0708	NaN	C

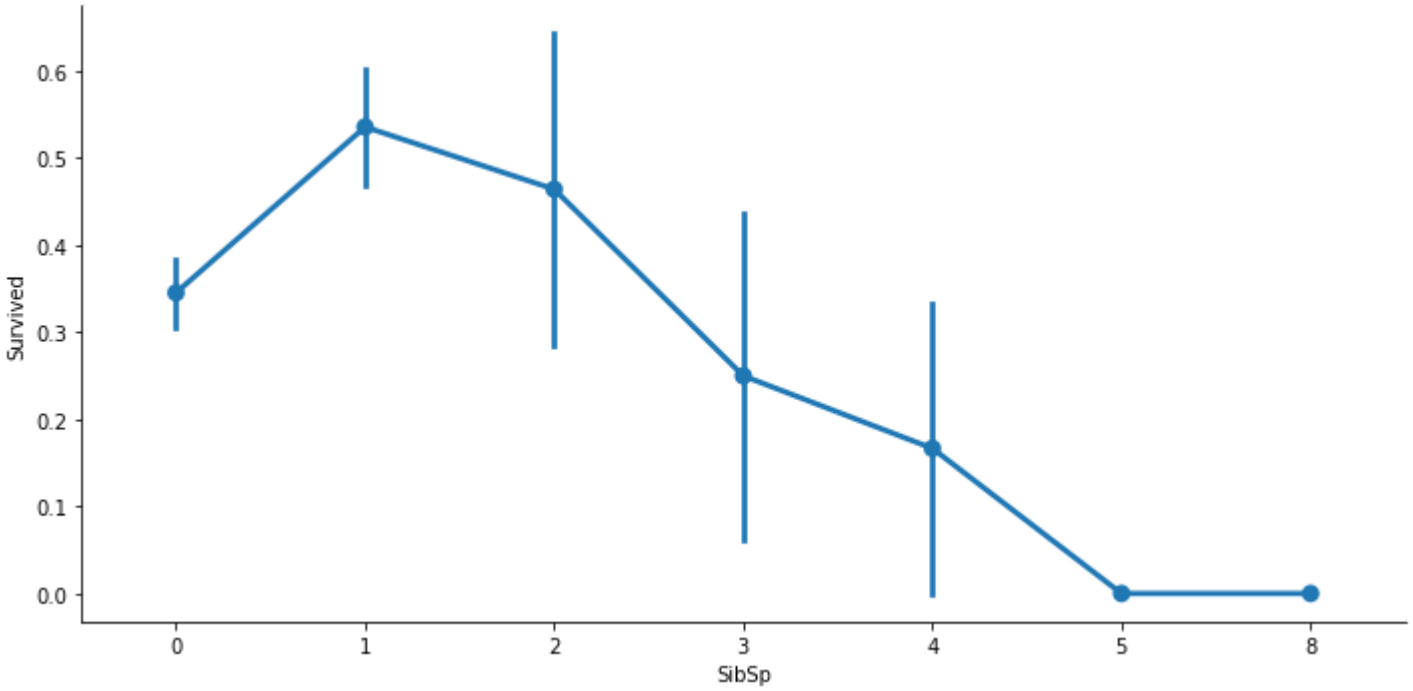
Combine SibSp & Parch

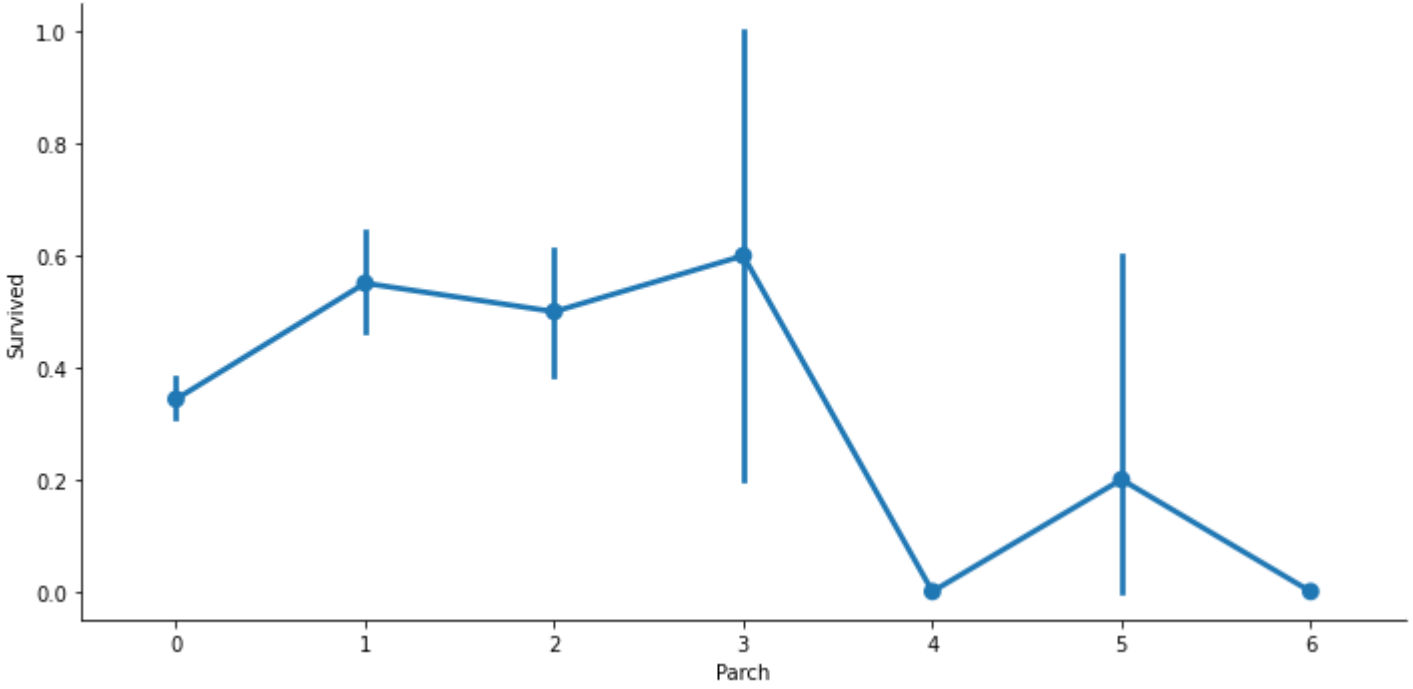
In [8]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline

for i, col in enumerate(['SibSp', 'Parch']):
    plt.figure(i)
    sns.catplot(x=col, y='Survived', data=titanic, kind='point', aspect=2, )
```

```
<Figure size 432x288 with 0 Axes>
```


Survived
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1
2
3
4
5
8
SibSp


Survived
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
Parch

In [9]:
```
titanic['Family_cnt'] = titanic['SibSp'] + titanic['Parch']
```

Drop unnnecessary variables

In [11]:
```
titanic.drop(['PassengerId', 'SibSp', 'Parch'], axis=1, inplace=True)
```

In [12]:
```
titanic.head()
```

Out[12]:

	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Family_cnt
0	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	7.2500	NaN	S	1
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	71.2833	C85	C	1
2	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	7.9250	NaN	S	0
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	53.1000	C123	S	1
4	0	3	Allen, Mr. William Henry	male	35.0	373450	8.0500	NaN	S	0

Clean categorical variables

Fill in missing & create indicator for `Cabin`

In [17]:
```
titanic.isnull().sum()
```

Out[17]:
```
Survived      0
Pclass        0
Name          0
Sex           0
Age           0
Ticket        0
Fare          0
Cabin       687
Embarked      2
Family_cnt    0
dtype: int64
```

In [20]:
```
titanic.groupby(titanic['Cabin'].isnull())['Survived'].mean()
```

Out[20]:
```
Cabin
False    0.666667
True     0.299854
Name: Survived, dtype: float64
```

In [21]:
```
titanic['Cabin_ind'] = np.where(titanic['Cabin'].isnull(), 0, 1)
titanic.head()
```

Out[21]:

	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Family_cnt	Cabin_ind
0	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	7.2500	NaN	S	1	0
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	71.2833	C85	C	1	1
2	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	7.9250	NaN	S	0	0
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	53.1000	C123	S	1	1
4	0	3	Allen, Mr. William Henry	male	35.0	373450	8.0500	NaN	S	0	0

Convert `Sex` to numeric

In [22]:
```
gender_num = {'male': 0, 'female': 1}

titanic['Sex'] = titanic['Sex'].map(gender_num)
titanic.head()
```

Out[22]:

	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Family_cnt	Cabin_ind
0	0	3	Braund, Mr. Owen Harris	0	22.0	A/5 21171	7.2500	NaN	S	1	0
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	PC 17599	71.2833	C85	C	1	1
2	1	3	Heikkinen, Miss. Laina	1	26.0	STON/O2. 3101282	7.9250	NaN	S	0	0
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	113803	53.1000	C123	S	1	1
4	0	3	Allen, Mr. William Henry	0	35.0	373450	8.0500	NaN	S	0	0

Drop unnecessary variables

In [23]:
```
titanic.drop(['Cabin', 'Embarked', 'Name', 'Ticket'], axis=1, inplace=True)
titanic.head()
```

Out[23]:

	Survived	Pclass	Sex	Age	Fare	Family_cnt	Cabin_ind
0	0	3	0	22.0	7.2500	1	0
1	1	1	1	38.0	71.2833	1	1
2	1	3	1	26.0	7.9250	0	0
3	1	1	1	35.0	53.1000	1	1
4	0	3	0	35.0	8.0500	0	0

Write out cleaned data

In [27]:
```
titanic.to_csv('./titanic_cleaned.csv')
```

Prepare Data: Split data into train, validation, and test set

Using the Titanic dataset from this Kaggle competition (we are only using the training set).

In this section, we will split the data into train, validation, and test set in preparation for fitting a basic model in the next section.

Read in Data

In [30]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

titanic = pd.read_csv('./titanic_cleaned.csv')
titanic.head()
```

Out[30]:

	Unnamed: 0	Survived	Pclass	Sex	Age	Fare	Family_cnt	Cabin_ind
0	0	0	3	0	22.0	7.2500	1	0

	Unnamed: 0	Survived	Pclass	Sex	Age	Fare	Family_cnt	Cabin_ind
1	1	1	1	1	38.0	71.2833	1	1
2	2	1	3	1	26.0	7.9250	0	0
3	3	1	1	1	35.0	53.1000	1	1
4	4	0	3	0	35.0	8.0500	0	0

Split into train, validation, and test set

In [31]:
```
features = titanic.drop('Survived', axis=1)
labels = titanic['Survived']

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.4, random_state = 42)
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size=0.5, random_state = 42)
```

In [32]:
```
for dataset in [y_train, y_val, y_test]:
    print(round(len(dataset) / len(labels), 2))
```

```
0.6
0.2
0.2
```

Write out all data

In [35]:
```
X_train.to_csv('./train_features.csv', index=False)
X_val.to_csv('./val_features.csv', index=False)
X_test.to_csv('./test_features.csv', index=False)

y_train.to_csv('./train_labels.csv', index=False)
y_val.to_csv('./val_labels.csv', index=False)
y_test.to_csv('./test_labels.csv', index=False)
```

Chapter 4

Boosting: Explore boosting algorithms in Python

Import `GradientBoostingClassifier` and `AdaBoostClassifier` from `sklearn` and explore the hyperparameters.

Import Boosting Algorithm for Classification

In [36]:
```
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier

GradientBoostingClassifier().get_params()
```

Out[36]:
```
{'ccp_alpha': 0.0,
 'criterion': 'friedman_mse',
 'init': None,
 'learning_rate': 0.1,
 'loss': 'deviance',
 'max_depth': 3,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_iter_no_change': None,
 'random_state': None,
 'subsample': 1.0,
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': 0,
 'warm_start': False}
```

In [37]:
```
AdaBoostClassifier().get_params()
```

Out[37]:
```
{'algorithm': 'SAMME.R',
 'base_estimator': None,
 'learning_rate': 1.0,
 'n_estimators': 50,
 'random_state': None}
```

Read in Data

In [39]:
```
import joblib
import pandas as pd
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

tr_features = pd.read_csv('./train_features.csv')
tr_labels = pd.read_csv('./train_labels.csv')
```

Hyperparameter tuning

In [41]:

```
def print_results(results):
    print('BEST PARAMS: {}\n'.format(results.best_params_))

    means = results.cv_results_['mean_test_score']
    stds = results.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, results.cv_results_['params']):
        print('{} (+/-{}) for {}'.format(round(mean, 3), round(std * 2, 3), params))
```

In [42]:

```
gb = GradientBoostingClassifier()
parameters = {
    'n_estimators': [5, 50, 250, 500],
    'max_depth': [1, 3, 5, 7, 9],
    'learning_rate': [0.01, 0.1, 1, 10, 100]
}
cv = GridSearchCV(gb, parameters, cv=5)
cv.fit(tr_features, tr_labels.values.ravel())

print_results(cv)
```

```
BEST PARAMS: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 500}

0.624 (+/-0.007) for {'learning_rate': 0.01, 'max_depth': 1, 'n_estimators': 5}
0.796 (+/-0.115) for {'learning_rate': 0.01, 'max_depth': 1, 'n_estimators': 50}
0.796 (+/-0.115) for {'learning_rate': 0.01, 'max_depth': 1, 'n_estimators': 250}
0.811 (+/-0.117) for {'learning_rate': 0.01, 'max_depth': 1, 'n_estimators': 500}
0.624 (+/-0.007) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 5}
0.811 (+/-0.069) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 50}
0.824 (+/-0.086) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 250}
0.828 (+/-0.074) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 500}
0.624 (+/-0.007) for {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 5}
0.809 (+/-0.046) for {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 50}
0.822 (+/-0.059) for {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 250}
0.816 (+/-0.042) for {'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 500}
0.624 (+/-0.007) for {'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 5}
```

```
0.817 (+/-0.053) for {'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 50}
0.807 (+/-0.026) for {'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 250}
0.798 (+/-0.022) for {'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 500}
0.624 (+/-0.007) for {'learning_rate': 0.01, 'max_depth': 9, 'n_estimators': 5}
0.798 (+/-0.049) for {'learning_rate': 0.01, 'max_depth': 9, 'n_estimators': 50}
0.787 (+/-0.036) for {'learning_rate': 0.01, 'max_depth': 9, 'n_estimators': 250}
0.773 (+/-0.021) for {'learning_rate': 0.01, 'max_depth': 9, 'n_estimators': 500}
0.796 (+/-0.115) for {'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 5}
0.815 (+/-0.119) for {'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 50}
0.818 (+/-0.109) for {'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 250}
0.818 (+/-0.118) for {'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 500}
0.813 (+/-0.071) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 5}
0.824 (+/-0.074) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 50}
0.803 (+/-0.032) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 250}
0.788 (+/-0.042) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 500}
0.811 (+/-0.061) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 5}
0.817 (+/-0.041) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 50}
0.798 (+/-0.048) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 250}
0.794 (+/-0.056) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 500}
0.822 (+/-0.056) for {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 5}
0.796 (+/-0.017) for {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 50}
0.802 (+/-0.036) for {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 250}
0.8 (+/-0.034) for {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 500}
0.798 (+/-0.052) for {'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 5}
0.798 (+/-0.045) for {'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 50}
0.796 (+/-0.041) for {'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 250}
0.798 (+/-0.039) for {'learning_rate': 0.1, 'max_depth': 9, 'n_estimators': 500}
0.818 (+/-0.099) for {'learning_rate': 1, 'max_depth': 1, 'n_estimators': 5}
0.818 (+/-0.114) for {'learning_rate': 1, 'max_depth': 1, 'n_estimators': 50}
0.815 (+/-0.069) for {'learning_rate': 1, 'max_depth': 1, 'n_estimators': 250}
0.79 (+/-0.08) for {'learning_rate': 1, 'max_depth': 1, 'n_estimators': 500}
0.813 (+/-0.065) for {'learning_rate': 1, 'max_depth': 3, 'n_estimators': 5}
0.788 (+/-0.069) for {'learning_rate': 1, 'max_depth': 3, 'n_estimators': 50}
0.802 (+/-0.047) for {'learning_rate': 1, 'max_depth': 3, 'n_estimators': 250}
0.807 (+/-0.053) for {'learning_rate': 1, 'max_depth': 3, 'n_estimators': 500}
0.775 (+/-0.049) for {'learning_rate': 1, 'max_depth': 5, 'n_estimators': 5}
0.798 (+/-0.067) for {'learning_rate': 1, 'max_depth': 5, 'n_estimators': 50}
0.807 (+/-0.057) for {'learning_rate': 1, 'max_depth': 5, 'n_estimators': 250}
0.803 (+/-0.047) for {'learning_rate': 1, 'max_depth': 5, 'n_estimators': 500}
0.766 (+/-0.013) for {'learning_rate': 1, 'max_depth': 7, 'n_estimators': 5}
0.798 (+/-0.041) for {'learning_rate': 1, 'max_depth': 7, 'n_estimators': 50}
0.792 (+/-0.027) for {'learning_rate': 1, 'max_depth': 7, 'n_estimators': 250}
0.772 (+/-0.045) for {'learning_rate': 1, 'max_depth': 7, 'n_estimators': 500}
0.783 (+/-0.049) for {'learning_rate': 1, 'max_depth': 9, 'n_estimators': 5}
0.794 (+/-0.058) for {'learning_rate': 1, 'max_depth': 9, 'n_estimators': 50}
```

```
0.783 (+/-0.04) for {'learning_rate': 1, 'max_depth': 9, 'n_estimators': 250}
0.766 (+/-0.026) for {'learning_rate': 1, 'max_depth': 9, 'n_estimators': 500}
0.204 (+/-0.115) for {'learning_rate': 10, 'max_depth': 1, 'n_estimators': 5}
0.204 (+/-0.115) for {'learning_rate': 10, 'max_depth': 1, 'n_estimators': 50}
0.204 (+/-0.115) for {'learning_rate': 10, 'max_depth': 1, 'n_estimators': 250}
0.204 (+/-0.115) for {'learning_rate': 10, 'max_depth': 1, 'n_estimators': 500}
0.369 (+/-0.374) for {'learning_rate': 10, 'max_depth': 3, 'n_estimators': 5}
0.369 (+/-0.374) for {'learning_rate': 10, 'max_depth': 3, 'n_estimators': 50}
0.369 (+/-0.374) for {'learning_rate': 10, 'max_depth': 3, 'n_estimators': 250}
0.369 (+/-0.374) for {'learning_rate': 10, 'max_depth': 3, 'n_estimators': 500}
0.489 (+/-0.15) for {'learning_rate': 10, 'max_depth': 5, 'n_estimators': 5}
0.489 (+/-0.156) for {'learning_rate': 10, 'max_depth': 5, 'n_estimators': 50}
0.494 (+/-0.154) for {'learning_rate': 10, 'max_depth': 5, 'n_estimators': 250}
0.49 (+/-0.145) for {'learning_rate': 10, 'max_depth': 5, 'n_estimators': 500}
0.607 (+/-0.172) for {'learning_rate': 10, 'max_depth': 7, 'n_estimators': 5}
0.605 (+/-0.187) for {'learning_rate': 10, 'max_depth': 7, 'n_estimators': 50}
0.62 (+/-0.141) for {'learning_rate': 10, 'max_depth': 7, 'n_estimators': 250}
0.59 (+/-0.204) for {'learning_rate': 10, 'max_depth': 7, 'n_estimators': 500}
0.706 (+/-0.103) for {'learning_rate': 10, 'max_depth': 9, 'n_estimators': 5}
0.715 (+/-0.162) for {'learning_rate': 10, 'max_depth': 9, 'n_estimators': 50}
0.732 (+/-0.13) for {'learning_rate': 10, 'max_depth': 9, 'n_estimators': 250}
0.721 (+/-0.105) for {'learning_rate': 10, 'max_depth': 9, 'n_estimators': 500}
0.376 (+/-0.007) for {'learning_rate': 100, 'max_depth': 1, 'n_estimators': 5}
0.376 (+/-0.007) for {'learning_rate': 100, 'max_depth': 1, 'n_estimators': 50}
0.376 (+/-0.007) for {'learning_rate': 100, 'max_depth': 1, 'n_estimators': 250}
0.376 (+/-0.007) for {'learning_rate': 100, 'max_depth': 1, 'n_estimators': 500}
0.281 (+/-0.118) for {'learning_rate': 100, 'max_depth': 3, 'n_estimators': 5}
0.281 (+/-0.118) for {'learning_rate': 100, 'max_depth': 3, 'n_estimators': 50}
0.281 (+/-0.118) for {'learning_rate': 100, 'max_depth': 3, 'n_estimators': 250}
0.281 (+/-0.118) for {'learning_rate': 100, 'max_depth': 3, 'n_estimators': 500}
0.47 (+/-0.192) for {'learning_rate': 100, 'max_depth': 5, 'n_estimators': 5}
0.468 (+/-0.188) for {'learning_rate': 100, 'max_depth': 5, 'n_estimators': 50}
0.472 (+/-0.178) for {'learning_rate': 100, 'max_depth': 5, 'n_estimators': 250}
0.475 (+/-0.185) for {'learning_rate': 100, 'max_depth': 5, 'n_estimators': 500}
0.565 (+/-0.11) for {'learning_rate': 100, 'max_depth': 7, 'n_estimators': 5}
0.563 (+/-0.129) for {'learning_rate': 100, 'max_depth': 7, 'n_estimators': 50}
0.584 (+/-0.164) for {'learning_rate': 100, 'max_depth': 7, 'n_estimators': 250}
0.539 (+/-0.114) for {'learning_rate': 100, 'max_depth': 7, 'n_estimators': 500}
0.646 (+/-0.12) for {'learning_rate': 100, 'max_depth': 9, 'n_estimators': 5}
0.646 (+/-0.082) for {'learning_rate': 100, 'max_depth': 9, 'n_estimators': 50}
0.588 (+/-0.203) for {'learning_rate': 100, 'max_depth': 9, 'n_estimators': 250}
0.61 (+/-0.187) for {'learning_rate': 100, 'max_depth': 9, 'n_estimators': 500}
```

In [43]:

```
cv.best_estimator_
```

Out[43]:
```
GradientBoostingClassifier(learning_rate=0.01, n_estimators=500)
```

Write out pickled model

In [44]:
```
joblib.dump(cv.best_estimator_, './models/GB_model.pkl')
```

Out[44]:
```
['./models/GB_model.pkl']
```

Chapter 5

Bagging: Explore bagging algorithms in Python

Import `RandomForestClassifier` from `sklearn` and explore the hyperparameters.

Import Random Forest Algorithm for Classification

In [45]:
```
from sklearn.ensemble import RandomForestClassifier

RandomForestClassifier().get_params()
```

Out[45]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 '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': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

In [47]:
```
import joblib
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

tr_features = pd.read_csv('./train_features.csv')
tr_labels = pd.read_csv('./train_labels.csv')
```

Hyperparameter tuning

In [49]:
```
def print_results(results):
    print('BEST PARAMS: {}\n'.format(results.best_params_))

    means = results.cv_results_['mean_test_score']
    stds = results.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, results.cv_results_['params']):
        print('{} (+/-{}) for {}'.format(round(mean, 3), round(std * 2, 3), params))
```

In [50]:
```
rf = RandomForestClassifier()
parameters = {
    'n_estimators': [5, 50, 250, 500],
    'max_depth': [4, 8, 16, 32, None]
}
cv = GridSearchCV(rf, parameters, cv=5)
cv.fit(tr_features, tr_labels.values.ravel())

print_results(cv)
```

```
BEST PARAMS: {'max_depth': 4, 'n_estimators': 500}

0.811 (+/-0.086) for {'max_depth': 4, 'n_estimators': 5}
0.818 (+/-0.119) for {'max_depth': 4, 'n_estimators': 50}
0.809 (+/-0.141) for {'max_depth': 4, 'n_estimators': 250}
0.822 (+/-0.103) for {'max_depth': 4, 'n_estimators': 500}
0.813 (+/-0.067) for {'max_depth': 8, 'n_estimators': 5}
0.818 (+/-0.107) for {'max_depth': 8, 'n_estimators': 50}
0.813 (+/-0.087) for {'max_depth': 8, 'n_estimators': 250}
0.813 (+/-0.101) for {'max_depth': 8, 'n_estimators': 500}
0.798 (+/-0.079) for {'max_depth': 16, 'n_estimators': 5}
0.815 (+/-0.073) for {'max_depth': 16, 'n_estimators': 50}
0.807 (+/-0.077) for {'max_depth': 16, 'n_estimators': 250}
```

```
0.805 (+/-0.075) for {'max_depth': 16, 'n_estimators': 500}
0.794 (+/-0.09) for {'max_depth': 32, 'n_estimators': 5}
0.807 (+/-0.081) for {'max_depth': 32, 'n_estimators': 50}
0.805 (+/-0.08) for {'max_depth': 32, 'n_estimators': 250}
0.813 (+/-0.069) for {'max_depth': 32, 'n_estimators': 500}
0.79 (+/-0.052) for {'max_depth': None, 'n_estimators': 5}
0.807 (+/-0.088) for {'max_depth': None, 'n_estimators': 50}
0.809 (+/-0.075) for {'max_depth': None, 'n_estimators': 250}
0.803 (+/-0.069) for {'max_depth': None, 'n_estimators': 500}
```

In [51]:
```
cv.best_estimator_
```

Out[51]:
```
RandomForestClassifier(max_depth=4, n_estimators=500)
```

Write out pickled model

In [53]:
```
joblib.dump(cv.best_estimator_, './models/RF_model.pkl')
```

Out[53]:
```
['./models/RF_model.pkl']
```

Chapter 6

Stacking: Explore stacking algorithms in Python

Import `StackingClassifier` from `sklearn` and explore the hyperparameters.

In [55]:
```
from sklearn.ensemble import StackingClassifier, GradientBoostingClassifier, RandomForestClassifier

estimators = [('gb', GradientBoostingClassifier()), ('rf', RandomForestClassifier())]
StackingClassifier(estimators = estimators).get_params()
```

Out[55]:
```
{'cv': None,
 'estimators': [('gb', GradientBoostingClassifier()),
  ('rf', RandomForestClassifier())],
 'final_estimator': None,
 'n_jobs': None,
 'passthrough': False,
 'stack_method': 'auto',
 'verbose': 0,
```

```
'gb': GradientBoostingClassifier(),
'rf': RandomForestClassifier(),
'gb__ccp_alpha': 0.0,
'gb__criterion': 'friedman_mse',
'gb__init': None,
'gb__learning_rate': 0.1,
'gb__loss': 'deviance',
'gb__max_depth': 3,
'gb__max_features': None,
'gb__max_leaf_nodes': None,
'gb__min_impurity_decrease': 0.0,
'gb__min_impurity_split': None,
'gb__min_samples_leaf': 1,
'gb__min_samples_split': 2,
'gb__min_weight_fraction_leaf': 0.0,
'gb__n_estimators': 100,
'gb__n_iter_no_change': None,
'gb__random_state': None,
'gb__subsample': 1.0,
'gb__tol': 0.0001,
'gb__validation_fraction': 0.1,
'gb__verbose': 0,
'gb__warm_start': False,
'rf__bootstrap': True,
'rf__ccp_alpha': 0.0,
'rf__class_weight': None,
'rf__criterion': 'gini',
'rf__max_depth': None,
'rf__max_features': 'auto',
'rf__max_leaf_nodes': None,
'rf__max_samples': None,
'rf__min_impurity_decrease': 0.0,
'rf__min_impurity_split': None,
'rf__min_samples_leaf': 1,
'rf__min_samples_split': 2,
'rf__min_weight_fraction_leaf': 0.0,
'rf__n_estimators': 100,
'rf__n_jobs': None,
'rf__oob_score': False,
'rf__random_state': None,
'rf__verbose': 0,
'rf__warm_start': False}
```

Read in Data

Hyperparameter tuning

In [56]:
```
def print_results(results):
    print('BEST PARAMS: {}\n'.format(results.best_params_))

    means = results.cv_results_['mean_test_score']
    stds = results.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, results.cv_results_['params']):
        print('{} (+/-{}) for {}'.format(round(mean, 3), round(std * 2, 3), params))
```

In [57]:
```
estimators = [('rf', RandomForestClassifier()),
              ('gb', GradientBoostingClassifier())]

sc = StackingClassifier(estimators=estimators)
sc.get_params()
```

Out[57]:
```
{'cv': None,
 'estimators': [('rf', RandomForestClassifier()),
  ('gb', GradientBoostingClassifier())],
 'final_estimator': None,
 'n_jobs': None,
 'passthrough': False,
 'stack_method': 'auto',
 'verbose': 0,
 'rf': RandomForestClassifier(),
 'gb': GradientBoostingClassifier(),
 'rf__bootstrap': True,
 'rf__ccp_alpha': 0.0,
 'rf__class_weight': None,
 'rf__criterion': 'gini',
 'rf__max_depth': None,
 'rf__max_features': 'auto',
 'rf__max_leaf_nodes': None,
 'rf__max_samples': None,
 'rf__min_impurity_decrease': 0.0,
 'rf__min_impurity_split': None,
 'rf__min_samples_leaf': 1,
 'rf__min_samples_split': 2,
 'rf__min_weight_fraction_leaf': 0.0,
 'rf__n_estimators': 100,
 'rf__n_jobs': None,
 'rf__oob_score': False,
 'rf__random_state': None,
```

```
'rf__verbose': 0,
'rf__warm_start': False,
'gb__ccp_alpha': 0.0,
'gb__criterion': 'friedman_mse',
'gb__init': None,
'gb__learning_rate': 0.1,
'gb__loss': 'deviance',
'gb__max_depth': 3,
'gb__max_features': None,
'gb__max_leaf_nodes': None,
'gb__min_impurity_decrease': 0.0,
'gb__min_impurity_split': None,
'gb__min_samples_leaf': 1,
'gb__min_samples_split': 2,
'gb__min_weight_fraction_leaf': 0.0,
'gb__n_estimators': 100,
'gb__n_iter_no_change': None,
'gb__random_state': None,
'gb__subsample': 1.0,
'gb__tol': 0.0001,
'gb__validation_fraction': 0.1,
'gb__verbose': 0,
'gb__warm_start': False}
```

In [59]:

```
import joblib
import pandas as pd
from sklearn.ensemble import StackingClassifier, GradientBoostingClassifier, RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

tr_features = pd.read_csv('./train_features.csv')
tr_labels = pd.read_csv('./train_labels.csv')
```

Hyperparameter tuning

In [60]:

```
def print_results(results):
    print('BEST PARAMS: {}\n'.format(results.best_params_))

    means = results.cv_results_['mean_test_score']
    stds = results.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, results.cv_results_['params']):
        print('{} (+/-{}) for {}'.format(round(mean, 3), round(std * 2, 3), params))
```

In [61]:
```
estimators = [('rf', RandomForestClassifier()),
              ('gb', GradientBoostingClassifier())]

sc = StackingClassifier(estimators=estimators)
sc.get_params()
```

Out[61]:
```
{'cv': None,
 'estimators': [('rf', RandomForestClassifier()),
  ('gb', GradientBoostingClassifier())],
 'final_estimator': None,
 'n_jobs': None,
 'passthrough': False,
 'stack_method': 'auto',
 'verbose': 0,
 'rf': RandomForestClassifier(),
 'gb': GradientBoostingClassifier(),
 'rf__bootstrap': True,
 'rf__ccp_alpha': 0.0,
 'rf__class_weight': None,
 'rf__criterion': 'gini',
 'rf__max_depth': None,
 'rf__max_features': 'auto',
 'rf__max_leaf_nodes': None,
 'rf__max_samples': None,
 'rf__min_impurity_decrease': 0.0,
 'rf__min_impurity_split': None,
 'rf__min_samples_leaf': 1,
 'rf__min_samples_split': 2,
 'rf__min_weight_fraction_leaf': 0.0,
 'rf__n_estimators': 100,
 'rf__n_jobs': None,
 'rf__oob_score': False,
 'rf__random_state': None,
 'rf__verbose': 0,
 'rf__warm_start': False,
 'gb__ccp_alpha': 0.0,
 'gb__criterion': 'friedman_mse',
 'gb__init': None,
 'gb__learning_rate': 0.1,
 'gb__loss': 'deviance',
 'gb__max_depth': 3,
 'gb__max_features': None,
 'gb__max_leaf_nodes': None,
 'gb__min_impurity_decrease': 0.0,
```

```
'gb__min_impurity_split': None,
'gb__min_samples_leaf': 1,
'gb__min_samples_split': 2,
'gb__min_weight_fraction_leaf': 0.0,
'gb__n_estimators': 100,
'gb__n_iter_no_change': None,
'gb__random_state': None,
'gb__subsample': 1.0,
'gb__tol': 0.0001,
'gb__validation_fraction': 0.1,
'gb__verbose': 0,
'gb__warm_start': False}
```

In [62]:

```
parameters = {
    'gb__n_estimators': [50, 100],
    'rf__n_estimators': [50, 100],
    'final_estimator': [LogisticRegression(C=0.1),
                        LogisticRegression(C=1),
                        LogisticRegression(C=10)],
    'passthrough': [True, False]
}
cv = GridSearchCV(sc, parameters, cv=5)
cv.fit(tr_features, tr_labels.values.ravel())

print_results(cv)
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
BEST PARAMS: {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 100, 'passthrough': True, 'rf__n_estimato
rs': 50}

0.822 (+/-0.13) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 50, 'passthrough': True, 'rf__n_es
timators': 50}
0.815 (+/-0.133) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 50, 'passthrough': True, 'rf__n_e
stimators': 100}
0.824 (+/-0.062) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 50, 'passthrough': False, 'rf__n_
estimators': 50}
0.824 (+/-0.062) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 50, 'passthrough': False, 'rf__n_
estimators': 100}
0.83 (+/-0.105) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 100, 'passthrough': True, 'rf__n_e
stimators': 50}
0.815 (+/-0.126) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 100, 'passthrough': True, 'rf__n_
estimators': 100}
0.82 (+/-0.058) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 100, 'passthrough': False, 'rf__n_
estimators': 50}
0.82 (+/-0.05) for {'final_estimator': LogisticRegression(C=0.1), 'gb__n_estimators': 100, 'passthrough': False, 'rf__n_e
stimators': 100}
0.817 (+/-0.099) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 50, 'passthrough': True, 'rf__n_est
imators': 50}
0.818 (+/-0.093) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 50, 'passthrough': True, 'rf__n_est
imators': 100}
0.818 (+/-0.071) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 50, 'passthrough': False, 'rf__n_es
timators': 50}
0.817 (+/-0.07) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 50, 'passthrough': False, 'rf__n_est
imators': 100}
0.824 (+/-0.095) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 100, 'passthrough': True, 'rf__n_es
timators': 50}
0.818 (+/-0.102) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 100, 'passthrough': True, 'rf__n_es
timators': 100}
0.82 (+/-0.061) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 100, 'passthrough': False, 'rf__n_es
timators': 50}
0.822 (+/-0.063) for {'final_estimator': LogisticRegression(C=1), 'gb__n_estimators': 100, 'passthrough': False, 'rf__n_e
stimators': 100}
0.815 (+/-0.088) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 50, 'passthrough': True, 'rf__n_es
timators': 50}
0.817 (+/-0.091) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 50, 'passthrough': True, 'rf__n_es
timators': 100}
0.822 (+/-0.061) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 50, 'passthrough': False, 'rf__n_e
stimators': 50}
0.817 (+/-0.069) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 50, 'passthrough': False, 'rf__n_e
stimators': 100}
0.824 (+/-0.082) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 100, 'passthrough': True, 'rf__n_e
stimators': 50}
```

```
0.82 (+/-0.083) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 100, 'passthrough': True, 'rf__n_es
timators': 100}
0.824 (+/-0.065) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 100, 'passthrough': False, 'rf__n_
estimators': 50}
0.818 (+/-0.064) for {'final_estimator': LogisticRegression(C=10), 'gb__n_estimators': 100, 'passthrough': False, 'rf__n_
estimators': 100}
S:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Write out pickled model

In [63]:
```
joblib.dump(cv.best_estimator_, './models/stacked_model.pkl')
```

Out[63]:
```
['./models/stacked_model.pkl']
```

Conclusion: Compare model results and final model selection

Using the Titanic dataset from this Kaggle competition.

In this section, we will do the following:

1. Evaluate all of our saved models on the validation set
2. Select the best model based on performance on the validation set
3. Evaluate that model on the holdout test set

Read in Data

In [66]:
```
import joblib
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score
from time import time
```

```
val_features = pd.read_csv('./val_features.csv')
val_labels = pd.read_csv('./val_labels.csv')

te_features = pd.read_csv('./test_features.csv')
te_labels = pd.read_csv('./test_labels.csv')
```

In [67]:
```
gb_mdl = joblib.load('./models/GB_model.pkl')
rf_mdl = joblib.load('./models/RF_model.pkl')
stacked_mdl = joblib.load('./models/stacked_model.pkl')
```

Evaluate models on the validation set

In [68]:
```
def evaluate_model(model, features, labels):
    start = time()
    pred = model.predict(features)
    end = time()
    accuracy = round(accuracy_score(labels, pred), 3)
    precision = round(precision_score(labels, pred), 3)
    recall = round(recall_score(labels, pred), 3)
    print('{} -- Accuracy: {} / Precision: {} / Recall: {} / Latency: {}ms'.format(str(model).split('(')[0],
                                                                                   accuracy,
                                                                                   precision,
                                                                                   recall,
                                                                                   round((end - start)*1000, 1)))
```

In [69]:
```
for mdl in [gb_mdl, rf_mdl, stacked_mdl]:
    evaluate_model(mdl, val_features, val_labels)
```

```
GradientBoostingClassifier -- Accuracy: 0.809 / Precision: 0.804 / Recall: 0.631 / Latency: 8.3ms
RandomForestClassifier -- Accuracy: 0.809 / Precision: 0.816 / Recall: 0.615 / Latency: 52.1ms
StackingClassifier -- Accuracy: 0.803 / Precision: 0.778 / Recall: 0.646 / Latency: 9.1ms
```

Evaluate best model on test set

In [70]:
```
evaluate_model(rf_mdl, te_features, te_labels)
```

```
RandomForestClassifier -- Accuracy: 0.799 / Precision: 0.87 / Recall: 0.618 / Latency: 54.0ms
```

In [ ]: