# JN

## Project - Titanic Dataset

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

```
# Load the training/test sets as pandas dataframes, assuming they're in same folder as
train_set = pd.read_csv('train.csv')
test_set = pd.read_csv('test.csv')

data_set = pd.concat([train_set, test_set], ignore_index=True) # combine the training,
data_set = data_set.reindex(columns=train_set.columns.values) # reorder columns to or:
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: FutureWarnin
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """
```

```
# verify data is combined, and row indices/column order are correct
data_set

#hide the dataset 0-->
```

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0.0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A |
| | | | | Cumings, Mrs. John | | | | | |

## Basic information of variables in the data set

PassengerId - ID # of the passenger

Survived - Whether the passenger survived or not, 1 for yes and 0 for no

Pclass - Ticket class of the passenger (1 = 1st, 2 = 2nd, 3 = 3rd)

Name, Sex, Age - Basic info of the passenger

SibSp - # of siblings/spouses of the passenger aboard the Titanic

ParCh - # of parents/children of the passenger aboard the Titanic

ticket - Ticket Number

Fare - Cost of ticket

Cabin - Cabin number of passenger

Embarked - Port which the passenger boarded the Titanic from (C = Cherbourg, Q = Queenstown, S = Southampton)

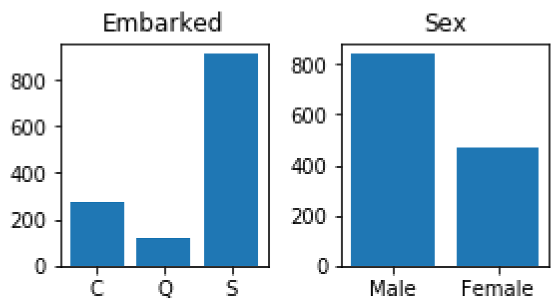
```
# Summary of variables listed in the data set
data_set.info()
```

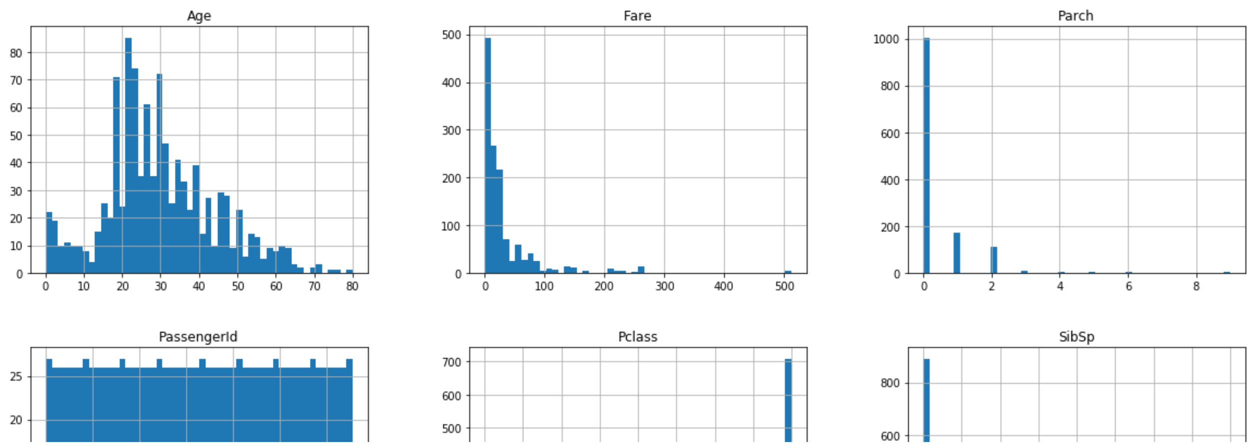
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
PassengerId    1309 non-null int64
Survived       891 non-null float64
Pclass         1309 non-null int64
Name           1309 non-null object
Sex            1309 non-null object
Age            1046 non-null float64
SibSp          1309 non-null int64
Parch          1309 non-null int64
Ticket         1309 non-null object
Fare           1308 non-null float64
Cabin          295 non-null object
Embarked       1307 non-null object
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

```
# Some graphical representations of some of the variables
```

```
# Embarked graph
embarked_x = ['C', 'Q', 'S']
embarked_y = [sum((data_set['Embarked']) == 'C'), sum((data_set['Embarked']) == 'Q'),
plt.subplot(231).set_title('Embarked')
plt.bar(embarked_x, embarked_y)
plt.tight_layout()

# Sex graph
sex_x = ['Male', 'Female']
sex_y = [sum((data_set['Sex']) == 'male'), sum((data_set['Sex']) == 'female')]
plt.subplot(232).set_title('Sex')
plt.bar(sex_x, sex_y)
plt.tight_layout()

data_set.hist(bins=50, figsize=(20,15))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1115b8690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x122ccac50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1233f1fd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x123425c90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1234664d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12349bcd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1234dc510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x123510d10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12351b890>]],
      dtype=object)
```




## Stage 1 - 4 - Data Cleansing

At first glance for the overall set, Cabin has the most null values, Survived is next, then Age. Fare and Embarked are only missing 1-2, which should be acceptable for our use case.

When looking at just the training set though (after dropping Cabin), Age has 177 values missing, and Embarked only has 2 values missing.

```
# for our use, Cabin will be dropped as most of it's values are null
# dropping Name and Ticket as well since they would be more complicated to turn into ı
# PassengerId also isn't of much use to us for train set, will leave in test set for I
# drop from train set, not dropping from test as we will be splitting the training set
train_set = train_set.drop(['Cabin', 'Name', 'Ticket', 'PassengerId'], axis=1)
train_set.info()

test_set = test_set.drop(['Cabin', 'Name', 'Ticket'], axis=1)
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
```

```
Data columns (total 8 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Sex         891 non-null object
Age         714 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
Embarked    889 non-null object
dtypes: float64(2), int64(4), object(2)
memory usage: 55.8+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Fare           417 non-null float64
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(2)
memory usage: 26.2+ KB
```

```
# filling null values of Age and Fare via Imputer

from sklearn.preprocessing import Imputer

# Save the two columns before temporarily dropping them
train_sex = train_set['Sex']
train_embarked = train_set['Embarked']

test_sex = test_set['Sex']
test_embarked = test_set['Embarked']

# drop the columns temporarily for imputer purposes
train_set = train_set.drop(['Sex', 'Embarked'], axis=1)
test_set = test_set.drop(['Sex', 'Embarked'], axis=1)
train_columns = train_set.columns.values
test_columns = test_set.columns.values

# take care of imputing the train set then adding the columns back
imputer_train = Imputer(strategy='median')
imputer_train.fit(train_set)

train_set = imputer_train.transform(train_set)
train_set = pd.DataFrame(train_set, columns=train_columns)
train_set['Sex'] = train_sex
train_set['Embarked'] = train_embarked

# impute the test set then add the columns back
imputer_test = Imputer(strategy='median')
```

```
imputer_test.fit(test_set)

test_set = imputer_test.transform(test_set)
test_set = pd.DataFrame(test_set, columns=test_columns)
test_set['Sex'] = test_sex
test_set['Embarked'] = test_embarked

train_set.info()
test_set.info()

# will take care of the 2 null values in training set for Embarked when that column is
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
Survived    891 non-null float64
Pclass      891 non-null float64
Age         891 non-null float64
SibSp       891 non-null float64
Parch       891 non-null float64
Fare        891 non-null float64
Sex         891 non-null object
Embarked    889 non-null object
dtypes: float64(6), object(2)
memory usage: 55.8+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
PassengerId    418 non-null float64
Pclass         418 non-null float64
Age            418 non-null float64
SibSp          418 non-null float64
Parch          418 non-null float64
Fare           418 non-null float64
Sex            418 non-null object
Embarked       418 non-null object
dtypes: float64(6), object(2)
memory usage: 26.2+ KB
/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/deprecation.py:66: Depre
  warnings.warn(msg, category=DeprecationWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/deprecation.py:66: Depre
  warnings.warn(msg, category=DeprecationWarning)
```

```
# Stage 2 - 1 Data Preparation
# Need to convert values in Sex columns to numbers. (0 = female, 1 = male)
train_set.replace('male', 1, inplace=True)
train_set.replace('female', 0, inplace=True)
test_set.replace('male', 1, inplace=True)
test_set.replace('female', 0, inplace=True)
train_set.info()
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
Survived    891 non-null float64
Pclass      891 non-null float64
Age         891 non-null float64
SibSp       891 non-null float64
Parch       891 non-null float64
Fare        891 non-null float64
Sex         891 non-null int64
Embarked    889 non-null object
dtypes: float64(6), int64(1), object(1)
memory usage: 55.8+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
PassengerId    418 non-null float64
Pclass         418 non-null float64
Age            418 non-null float64
SibSp          418 non-null float64
Parch          418 non-null float64
Fare           418 non-null float64
Sex            418 non-null int64
Embarked       418 non-null object
dtypes: float64(6), int64(1), object(1)
memory usage: 26.2+ KB
```

```
train_set['Embarked'].value_counts() # find most common value for Embarked in test to
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

```
# convert Embarked column into 3 numeric columns (Embark_S, Embark_C, Embark_Q)
# train set first
train_set['Embarked'].fillna('S', inplace=True) # fill null values of Embarked with S
embark_num = pd.get_dummies(train_set['Embarked'], prefix='Embark')
train_set['Embark_C'] = embark_num['Embark_C']
train_set['Embark_Q'] = embark_num['Embark_Q']
train_set['Embark_S'] = embark_num['Embark_S']

# now for test set
embark_num = pd.get_dummies(test_set['Embarked'], prefix='Embark')
test_set['Embark_C'] = embark_num['Embark_C']
test_set['Embark_Q'] = embark_num['Embark_Q']
test_set['Embark_S'] = embark_num['Embark_S']

train_set.head() # verify the new columns before dropping Embarked
```

| | Survived | Pclass | Age | SibSp | Parch | Fare | Sex | Embarked | Embark_C | Emb |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 3.0 | 22.0 | 1.0 | 0.0 | 7.2500 | 1 | S | 0 | |
| 1 | 1.0 | 1.0 | 38.0 | 1.0 | 0.0 | 71.2833 | 0 | C | 1 | |
| 2 | 1.0 | 3.0 | 26.0 | 0.0 | 0.0 | 7.9250 | 0 | S | 0 | |

```
test_set.head() # verify test set as well
```

| | PassengerId | Pclass | Age | SibSp | Parch | Fare | Sex | Embarked | Embark_C |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 892.0 | 3.0 | 34.5 | 0.0 | 0.0 | 7.8292 | 1 | Q | 0 |
| 1 | 893.0 | 3.0 | 47.0 | 1.0 | 0.0 | 7.0000 | 0 | S | 0 |
| 2 | 894.0 | 2.0 | 62.0 | 0.0 | 0.0 | 9.6875 | 1 | Q | 0 |
| 3 | 895.0 | 3.0 | 27.0 | 0.0 | 0.0 | 8.6625 | 1 | S | 0 |
| 4 | 896.0 | 3.0 | 22.0 | 1.0 | 1.0 | 12.2875 | 0 | S | 0 |

```
# drop Embarked from both sets now
train_set = train_set.drop(['Embarked'], axis=1)
train_set.info()

test_set = test_set.drop(['Embarked'], axis=1)
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
Survived    891 non-null float64
Pclass      891 non-null float64
Age         891 non-null float64
SibSp       891 non-null float64
Parch       891 non-null float64
Fare        891 non-null float64
Sex         891 non-null int64
Embark_C    891 non-null uint8
Embark_Q    891 non-null uint8
Embark_S    891 non-null uint8
dtypes: float64(6), int64(1), uint8(3)
memory usage: 51.5 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 10 columns):
PassengerId    418 non-null float64
Pclass         418 non-null float64
Age            418 non-null float64
SibSp          418 non-null float64
Parch          418 non-null float64
Fare           418 non-null float64
Sex            418 non-null int64
```

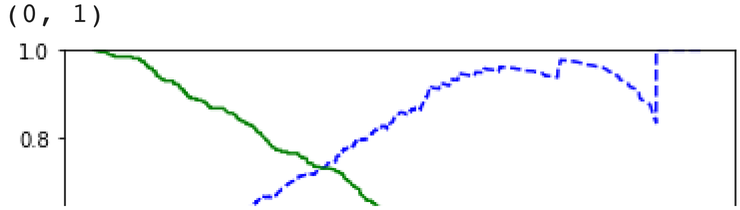
```
Embark_C       418 non-null uint8
Embark_Q       418 non-null uint8
Embark_S       418 non-null uint8
dtypes: float64(6), int64(1), uint8(3)
memory usage: 24.2 KB
```

```
# Feature scaling - only scaled age and fare in train set currently, maybe try pclass
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
column_order = train_set.columns.values

# Test scaling on age since there is a large range for this column
age_std = scaler.fit_transform(train_set['Age'].values.reshape(891,1))
train_set = train_set.drop(['Age'], axis=1)
train_set['Age'] = age_std

fare_std = scaler.fit_transform(train_set['Fare'].values.reshape(891,1))
train_set = train_set.drop(['Fare'], axis=1)
train_set['Fare'] = fare_std

train_set = train_set.reindex(columns=column_order) # reorder train set columns after
train_set.info()
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
Survived    891 non-null float64
Pclass      891 non-null float64
Age         891 non-null float64
SibSp       891 non-null float64
Parch       891 non-null float64
Fare        891 non-null float64
Sex         891 non-null int64
Embark_C    891 non-null uint8
Embark_Q    891 non-null uint8
Embark_S    891 non-null uint8
dtypes: float64(6), int64(1), uint8(3)
memory usage: 51.5 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 10 columns):
PassengerId    418 non-null float64
Pclass         418 non-null float64
Age            418 non-null float64
SibSp          418 non-null float64
Parch          418 non-null float64
Fare           418 non-null float64
Sex            418 non-null int64
Embark_C       418 non-null uint8
Embark_Q       418 non-null uint8
Embark_S       418 non-null uint8
dtypes: float64(6), int64(1), uint8(3)
memory usage: 24.2 KB
```

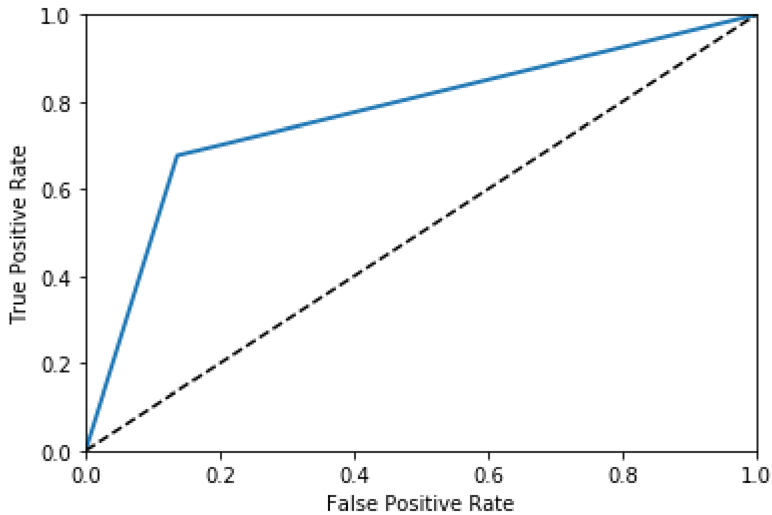
```
# Train/validation split
from sklearn.model_selection import train_test_split
train_set_X, val_set = train_test_split(train_set, test_size=0.20)
X = train_set_X.iloc[:, 1:10]
y = train_set_X.iloc[:, 0]
val_X = val_set.iloc[:, 1:10]
val_y = val_set.iloc[:, 0]


# Test with Logistic Regression

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.metrics import accuracy_score, precision_score, confusion_matrix, recall_

log_reg = LogisticRegression(n_jobs=-1, random_state=28245)
log_reg.fit(X, y)

# cross validation
cross_val = cross_val_score(log_reg, X, y, cv=5)
print('Cross Validation Mean:', cross_val.mean())

# test accuracy score with validation set
test_pred = log_reg.predict(val_X)
print("Test Accuracy:", accuracy_score(val_y, test_pred))

# confusion matrix/precision/recall
print("Confusion Matrix:", confusion_matrix(val_y, test_pred))
print("Precision Score:", precision_score(val_y, test_pred))
print("Recall Score:", recall_score(val_y, test_pred))

# precision/recall tradeoff
y_probas = cross_val_predict(log_reg, X, y, cv=5, method='decision_function')
precisions, recalls, thresholds = precision_recall_curve(y, y_probas)
plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
plt.xlabel("Threshold")
plt.legend(loc="lower left")
plt.ylim([0, 1])
```

```
Cross Validation Mean: 0.808913621589678
Test Accuracy: 0.7932960893854749
Confusion Matrix: [[93 14]
 [23 49]]
Precision Score: 0.7777777777777778
Recall Score: 0.6805555555555556
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py
  " = {}.".format(effective_n_jobs(self.n_jobs)))
(0, 1)
```



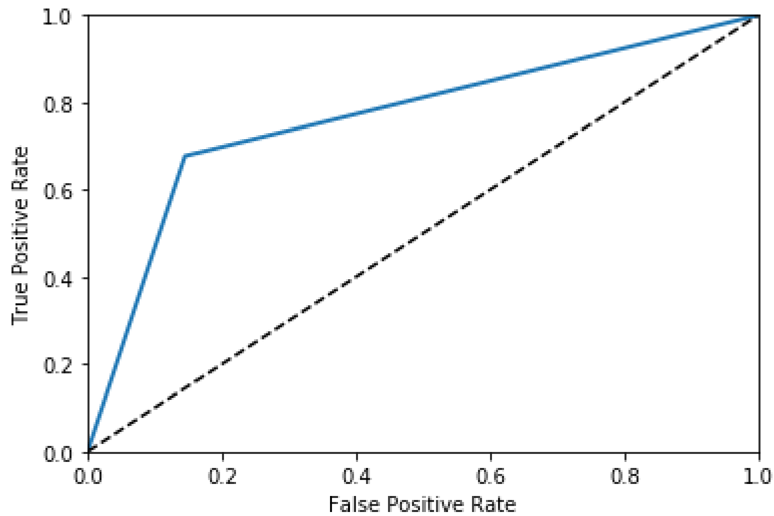
```
# ROC curve for Logistic Regression
fpr, tpr, thresholds = roc_curve(val_y, test_pred)
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0, 1], [0, 1], 'k--')
plt.axis([0, 1, 0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
Text(0, 0.5, 'True Positive Rate')
```




```
# KNN test
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X, y)

# cross validation
cross_val = cross_val_score(knn, X, y, cv=5)
print('Cross Validation Mean:', cross_val.mean())

# test accuracy score with validation set
test_pred_knn = knn.predict(val_X)
print("Test Accuracy:", accuracy_score(val_y, test_pred_knn))

# confusion matrix/precision/recall
print("Confusion Matrix:", confusion_matrix(val_y, test_pred_knn))
print("Precision Score:", precision_score(val_y, test_pred_knn))
print("Recall Score:", recall_score(val_y, test_pred_knn))

# in the case of KNN, a precision/recall tradeoff curve doesn't serve much purpose, a:

# ROC curve
fpr, tpr, thresholds = roc_curve(val_y, test_pred_knn)
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0, 1], [0, 1], 'k--')
plt.axis([0, 1, 0, 1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
Cross Validation Mean: 0.7963262090022654
Test Accuracy: 0.7932960893854749
Confusion Matrix: [[100  17]
 [ 20  42]]
Precision Score: 0.711864406779661
Recall Score: 0.6774193548387096
Text(0, 0.5, 'True Positive Rate')
```



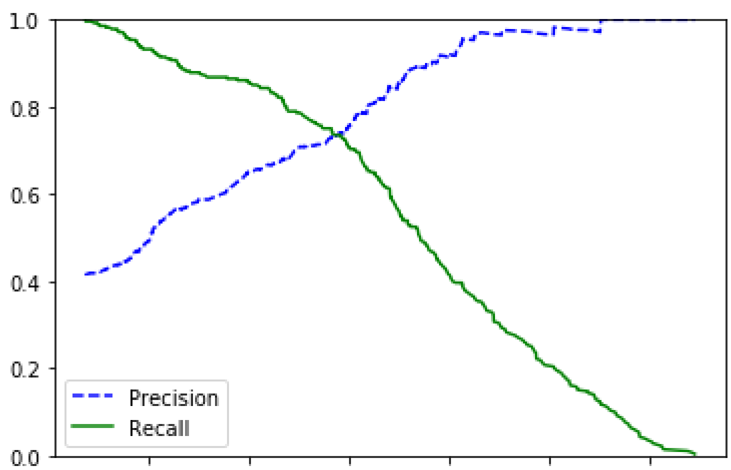

```
# Logistic Regression via GridSearch
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
log_reg_grid = LogisticRegression()
grid_search = GridSearchCV(\
                     estimator=log_reg_grid,
                     param_grid=[{'penalty' : ['l1', 'l2']}, {'solver' : ['newton-cg',
                                  {'max_iter' : range(100, 1000)}], scoring='accuracy',
                                 n_jobs=-1)

grid_search.fit(X, y)

best_model = grid_search.best_estimator_

# cross validation
cross_val = cross_val_score(best_model, X, y, cv=10, scoring='accuracy')
print('Cross Validation Mean:', cross_val.mean())

# test accuracy score with validation set
test_pred_gs = best_model.predict(val_X)
print("Test Accuracy:", accuracy_score(val_y, test_pred_gs))

# confusion matrix/precision/recall
print("Confusion Matrix:", confusion_matrix(val_y, test_pred_gs))
print("Precision Score:", precision_score(val_y, test_pred_gs))
print("Recall Score:", recall_score(val_y, test_pred_gs))

# precision/recall tradeoff
y_probas = cross_val_predict(log_reg_grid, X, y, cv=5, method='decision_function')
precisions, recalls, thresholds = precision_recall_curve(y, y_probas)
```

```
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.xlabel("Threshold")
    plt.legend(loc="lower left")
    plt.ylim([0, 1])
```

```
Cross Validation Mean: 0.7962832550860719
Test Accuracy: 0.7988826815642458
Confusion Matrix: [[101  16]
 [ 20  42]]
Precision Score: 0.7241379310344828
Recall Score: 0.6774193548387096
```

```
# Logistic Regression via RandomizedSearch
log_reg_random = LogisticRegression()

C_range = np.random.normal(1, 0.2, 10).astype(float)
C_range[C_range < 0] = 0.0001

params = {'penalty': ['l1', 'l2'],
         'C': C_range}
random_search = RandomizedSearchCV(\
                                    log_reg_random,
                                     params,
                                     n_iter=100,
                                     cv=5,
                                     n_jobs=-1,
                                     random_state=329438
                                    )
random_search.fit(X, y)

best_model = random_search.best_estimator_

# cross validation
cross_val = cross_val_score(best_model, X, y, cv=10, scoring='accuracy')
print('Cross Validation Mean:', cross_val.mean())

# test accuracy score with validation set
test_pred_rand = best_model.predict(val_X)
print("Test Accuracy:", accuracy_score(val_y, test_pred_rand))

# confusion matrix/precision/recall
print("Confusion Matrix:", confusion_matrix(val_y, test_pred_rand))
print("Precision Score:", precision_score(val_y, test_pred_rand))
print("Recall Score:", recall_score(val_y, test_pred_rand))

# precision/recall tradeoff
y_probas = cross_val_predict(log_reg_random, X, y, cv=5, method='decision_function')
precisions, recalls, thresholds = precision_recall_curve(y, y_probas)
plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
plt.xlabel("Threshold")
plt.legend(loc="lower left")
plt.ylim([0, 1])
```

```
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  % (grid_size, self.n_iter, grid_size), UserWarning)
Cross Validation Mean: 0.7948552425665102
Test Accuracy: 0.7988826815642458
Confusion Matrix: [[101  16]
 [ 20  42]]
Precision Score: 0.7241379310344828
Recall Score: 0.6774193548387096
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
C:\Users\nbonet\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn
  FutureWarning)
(0, 1)
```




Double-click (or enter) to edit