**Module 12**

**Classification and K-Nearest Neighbors**

* [Video Transcripts](https://student.emeritus.org/courses/4765/files/3296371?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/3296371/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/3296372?wrap=1)

**Accuracy**

The ratio of correctly predicted observations to total observations

**Confusion Matrix**

A table that is often used to describe the performance of a classification model

**Decision Boundary**

A boundary that separates the data points into specific classes where the algorithm switches from one class to the next

**F1**

The weighted average of both precision and recall

**K-Nearest Neighbors**

An approach to data classification that estimates how likely a data point is to be a member of one group or another depending on which group the data points nearest to it are in

**Precision**

The proportion of accurately predicted positive observations in relation to the total predicted positive observations

**Recall**

The proportion of correctly predicted positive observations in relation to all of the observations in an actual class

**ROC Curve**

A graphical plot used to show the diagnostic ability of binary classifiers

Before focusing specifically on classification, let us review the different types of problems you have tackled so far and compare them against the classification task.

**Clustering (Module 6)**

Clustering is an unsupervised learning model that groups a population or set of data points in such a way that data points in the same group are similar to each other. In Module 6, you used the k-means algorithm to perform clustering. The goal of k-means is to group data points into k-clusters based on similarity or closeness between clusters.

**Regression (Module 7)**

Regression is a supervised learning model that predicts the relationship between a dependent variable and one or more independent variables. Regression, at its core, is simply comparing one variable with another variable and finding the relationship between them. Regression also allows you to observe how strongly one variable influences another. Linear regression, specifically, assumes that the relationship between variables can be plotted using a straight line.

**Time Series (Module 10)**

Time series can be a supervised learning or unsupervised learning model that orders a series of data points according to time. This method utilizes time as an independent variable, and its goal is to make a future forecast. Analysts use time series analysis to record data points at consistent intervals over a set duration that can then be used to model, simulate, and forecast behavior and inform strategic decision-making. Time series has many practical applications in business, economics, and finance, but it can be applied to any industry that compiles consistent historical data for analysis.

**Classification (Module 12)**

Classification is a supervised learning model that involves predicting the value of a target variable by building a model based on one or more predictor variables. Unlike regression, where the goal is to find a real-valued outcome, classification asks a completely different question. Classification models look at some specific features (predictors) in the data and then predict the value of a target variable. In other words, classification models ask you to predict what class a sample belongs to given a set of features.

Examine the table below to learn how classification compares to the other methods mentioned in this mini-lesson.

An overview of problem identification methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Classification** | **Clustering** | **Regression** | **Time Series** |
| **Definition** | Predicts the value of a target variable by building a model based on one or more predictors | Groups a population or set of data points | Predicts the relationship between the dependent variable and one or more independent variables | Utilizes time as an independent variable whose goal is to make a future forecast |
| **Goal** | Compute the category of data | Group similar items into clusters | Forecast or predict | Future forecast |
| **Real-World Applications** | Email spam protection, customer churn, conversion prediction | Fraud detection, similarity pattern detection, genetics | Predicting stock market prices, insights on customer behavior, marketing effectiveness | Seasonality, weather forecast, financial market trend analysis |
| **Algorithm** | K-Nearest Neighbors,  decision trees | K-Means | Linear regression | ARMA |

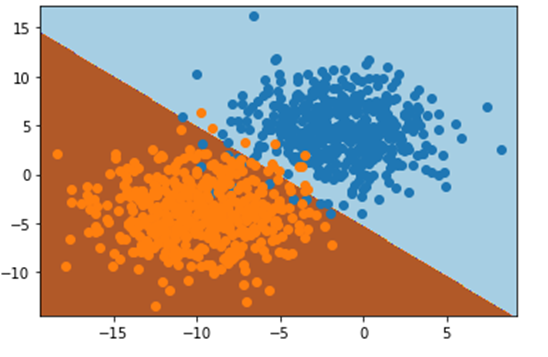
**Notes:**

the KNN algorithm finds the distance between a given data point and k numbers of other points in the dataset that are close to the initial point. The algorithm then votes for the most prevalent category for each individual point. Selecting a lower value may result in overfitting, while choosing a higher value may require high computational complexity.

To train a classifier on a dataset, you must define a set of hyperplanes. These hyperplanes are called decision boundaries, and they separate the data points into specific classes where the algorithm switches from one class to the next. A data point is more likely to be classified as class A on one side of a boundary and class B on the other.

In the logistic regression example below, a decision boundary is a straight line that separates class A from class B. However, it is difficult in linear models to determine the exact boundary line separating the two classes, so points from class A have also come into the region of class B.

Visualizing decision boundaries in this manner helps demonstrate how sensitive models are to the specific dataset, which can help understand how particular algorithms work and their limitations.



Classification performance is measured either by a numeric metric, like accuracy, or a graphical representation, like a receiver operating characteristic (ROC) curve. Classification metrics are based on the true positives (TPs), false positives (FPs), false negatives (FNs), and true negatives (TNs) contained in the confusion matrix.

A confusion matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Predicted |  |
|  |  | **+** | **−** |
| **Actual** | **+** | TP | FN  Type II error |
|  | **−** | FP  Type I error | TN |

**Accuracy**

Accuracy is the most intuitive measure of performance, as it is simply the ratio of correctly predicted observations to total observations. Accuracy can be deceiving in that it may signal a highly accurate model, but in all actuality, it has some weaknesses. Accuracy is only useful when the dataset is perfectly symmetrical, where values of false negatives and false positives are almost identical with similar costs.

**Precision**

Precision is the proportion of accurately predicted positive observations in relation to the total predicted positive observations. High precision is directly correlated to a low false-positive rate.

**Recall**

Recall (a.k.a. sensitivity) is the proportion of correctly predicted positive observations in relation to all of the observations in an actual class. As a result, recall measures the precision with which our model can determine the relevant data.

**F1**

F1 is the weighted average of both precision and recall.

A table outlining the metrics that are typically used to determine the performance of a model

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **When to Use** | **Formula** | **Example** |
| **Accuracy** | Used when you have a perfectly symmetrical dataset | (TP + TN) ÷ (TP + FP + FN + TN) | One out of every ten labels is incorrect, and nine are correct. Therefore, the accuracy is 0.90. |
| **Precision** | Used when you want to be more confident of your true positives | TP ÷ (TP + FP) | Two out of every ten cancer samples labeled by our program are healthy, and eight are cancerous. Therefore, the precision is 0.80. |
| **Recall** | Used when the idea of false positives is far better than false negatives | TP ÷ (TP + FN) | Three out of every ten COVID-19 patients are mislabeled by our program as negative, and seven are labeled as positive. Therefore, the recall is 0.70. |
| **F1** | Used when you have uneven class distribution | 2 × (Recall × Precision) ÷ (Recall + Precision) | Four out of every ten healthy people are mislabeled as having COVID-19, and six are correctly labeled as healthy. Therefore, the recall is 0.60. |

Example: false positive (classified malignant but benign) or false negative (classified as benign but malignant)

**Savio’s session:**

KNN algorithm: n\_neighbors =5, weights = uniform or distance

GridSearchCV internal does cross validation, so, pass the entire dataset

```python

from sklearn.inspection import PartialDependenceDisplay, partial\_dependence

fig, ax = plt.subplots(figsize = (20, 6))

PartialDependenceDisplay.from\_estimator(pipe, X, features = ['hdlngth', 'totlngth', 'footlgth', 'earconch', 'eye', 'chest'], ax = ax)

ax.set\_title('Partial Dependence Plots for 6 Features')

```

**Module Issues:**

Codio 12.3 Problem 2: the solution expects 'preds' set to base\_pipe.predict(X\_test)

Codio 12.4 Problem 5: best\_score is supposed to set

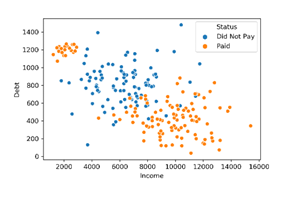
Codio 12.4 Problem 6: variable is supposed to be fn

Codio 12.6 Problem 6: scoring parameter is supposed to be roc\_auc

**Quizes:**

The goal of classification is to predict a real-valued outcome given a set of features. : False

*You are correct! The answer “*False*” is correct because the goal of classification is to predict what class a sample belongs to.*



In the given plot, the customer’s income is on the x-axis and the customer’s total debt is on the y-axis. Suppose there is a new customer who has an income of 13,000 and a total debt of 400. Which class would the customer belong to? : Paid

*You are correct! The answer “*Paid*” is correct because when the new data point is plotted on the graph, it is in the cluster of Paid data points, so it is quite obvious that it belongs to the Paid class.*

Which of the following is not a classification model? : Linear regression

*You are correct! The answer “*Linear regression*” is correct because classification is about predicting a label, and regression is about predicting a quantity.*

The nearest neighbor classifier simply asks: “Which data point in the training set is (blank) to the data point about which the prediction is being made?” : closest

*You are correct! The answer “*closest*” is correct because the nearest neighbors classifier finds the closest data point in the original data to the one you are trying to classify.*

Which one of the following is a constructor for the “KNeighborsClassifier” object? : n\_neighbors = “value”

*You are correct! The answer “*n\_neighbors = “value”*” is correct because this is the constructor for the “KNeighborsClassifier” object.*

In a nearest neighbor classifier, maximum complexity is k = 1, and minimum complexity is k = N. : True

*You are correct! The answer “*True*” is correct because for k = N, the model simply returns the most common class from the training set. By contrast, for k = 1, the model has to find exactly which training data point in the entire set is closest and then return exactly that data point.*

The mean squared error (MSE) of a classification model can be used as a metric for model quality. : False

*You are correct! The answer “*False*” is correct because in classification models, predictions and observations are not numerical quantities. Since they are not numerical, the squared difference between the prediction and observation cannot be computed.*

For a small dataset, suppose four out of five predictions are correct. What would the misclassification rate be? : 20% (1-accuracy)

*You are correct! The answer “*20%*” is correct because the accuracy of the prediction is 0.8 and the misclassification rate is given as “1****−****accuracy”, so it would be 0.2 in this scenario, which is 20%.*

For a nearest neighbors classifier, given *k* = 1, the misclassification rate on the training set is 0. : True

*You are correct! The answer “*True*” is correct because the closest data point to itself is itself, and thus the prediction for any training point will always be correct since the classifier will always use the training point’s own class as its prediction.*

A nearest neighbors model is built with different values of *k*, and the misclassification error is computed on each value of *k*. On what basis is the best possible *k* selected? : Least misclassification error

*You are correct! The answer “*Least misclassification error*” is correct because selecting the optimal*k*value for a nearest neighbors model means choosing the one with the least error. For classification models, error is computed as misclassification error.*

Scikit-learn models provide a function called predict\_proba, which returns not just predictions about the class of a given sample but also information about the level of confidence of the model. : True

*You are correct! The answer “*True*” is correct because the function tells you about the prediction class and the level of confidence interval.*

Assume there are two classes, “paid” and “did not pay.” Which class would be considered the 0th class in the predict\_proba() function on the nearest neighbors model object? : did not pay

Y*ou are correct! The answer “*did not pay*” is correct because the function selects the 0th class on the basis of alphabetical order. Since “did not pay” is the class that comes earlier in alphabetical order, it will be the 0th class.*

If a given nearest neighbor model has k=10, for the function predict\_proba() the probabilities will be a multiple of (blank). : 0.1

*You are correct! The answer “*0.1*” is correct because the behavior of each available training sample is used as a vote. Since we have ten nearest neighbors, our probabilities are always multiples of 0.1.*

For a binary classifier, scikit-learn returns class 1 if 𝑝≤0.5 and class 0 if 𝑝>0.5. : False

*You are correct! The answer “*False*” is correct because for a binary classifier, scikit-learn returns class 0 if 𝑝≤0.5 and class 1 if 𝑝>0.5.*

If you have imbalanced classes in your data, the accuracy measure is the most suitable classification metric. : False

*You are correct! The answer “*False*” is correct because in the case of imbalanced classes, accuracy does not necessarily give a good assessment of the usefulness of a model.*

What is a 2x2 table used as a classification metric known as? : Confusion matrix

*You are correct! The answer “*Confusion matrix*” is correct because this 2x2 table used as a classification metric is known as confusion matrix.*

A confusion matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **True** | 0 | 349 | **7** |
|  | 1 | **118** | **126** |
|  |  | 0 | 1 |
|  |  | **Predicted** |  |

In the given confusion matrix, what is the measure of true positive? : 126

*You are correct! The answer “*126*” is correct because true positive is an outcome where the model correctly predicts the positive class, which is the bottom right block where the condition is 1 and the prediction is 1 as well.*

What is the formula for accuracy using a confusion matrix? : (TP + TN) ÷ (TN + FP + FN + TP)

*You are correct! The answer “*(TP + TN) ÷ (TN + FP + FN + TP)*” is correct because this is the correct formula for accuracy using a confusion matrix.*

A confusion matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **True** | 0 | 349 | **7** |
|  | 1 | **118** | **126** |
|  |  | 0 | 1 |
|  |  | **Predicted** |  |

In the given confusion matrix, what is the measure of recall? : 51.6%

*You are correct! The answer “*51.6%*” is correct because the recall is the measure of the model correctly identifying true positives with formula TP ÷ (TP + FN), which comes out to be 126 ÷ (126 + 118) = 51.6.*

In a *k*-nearest neighbors model, when the threshold (T) is increased, precision gets better and recall gets worse. : True

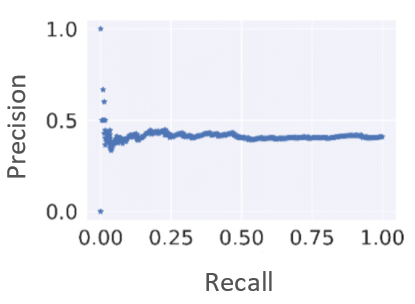
*You are correct! The answer “*True*” is correct because with the increase in T, the model only classifies data as 1 if it is very sure, which improves precision, and the model will classify fewer and fewer data as class 1, which worsens the recall.*

Scikit-learn provides a function that generates precision and recall values for various thresholds for a classifier. What is this function called? : precision\_recall\_curve()

*You are correct! The answer “*precision\_recall\_curve()*” is correct because this is the function used in scikit-learn to generate precision and recall values for various thresholds.*

In a hypothetical scenario, a model is built for bomb detection. Which metric is the most important? : Recall

*You are correct! The answer “*recall*” is correct because a failure to detect a bomb would be catastrophic, whereas a modest false positive rate would not be too disastrous for the system.*



Which type of model shows a precision recall curve like the one given in the plot? : No skill model

*You are correct! The answer “*No skill model*” is correct because such a model has no predictive skill. On any dataset, the no skill model will roughly approximate a horizontal line with precision equal to the fraction of items that are in the positive class.*

What is a receiver operator characteristic (ROC) curve? : The curve between recall and 1-specificity, The curve between the true positive rate and the true negative rate

*You are correct! The answers “*The curve between the true positive rate and the true negative rate*” and “*The curve between recall and 1-specificity*” are correct because the tradeoff between these measures is known as the ROC curve.*

What is the function in scikit-learn used for solving regression problems with nearest neighbors? : KNeighborsRegressor()

*You are correct! The answer “*KNeighborsRegressor()*” is correct because this is the function and library of scikit-learn used for solving regression problems with nearest neighbors.*

The output of the predict() function for KNearestRegressor() is a label. : False

*You are correct! The answer “*False*” is correct because the output of the predict() function for KNearestRegressor() model is a numerical value.*

Which metric should be used as cross-validation for KNearestRegressor to find the optimal value of K? : Mean squared error

*You are correct! The answer “*Mean squared error*” is correct because this is the scoring metric used in a regression problem.*

**Try-It Activity 12.1: Choosing the Right Metric**

**Blood Transfusion Service Center Dataset**

This is a donor dataset whom donated blood over time per "Blood Transfusion Service Center" where a bus visits a university for blood drive in Hsin-Chu City in Taiwan every 3 months, here: <https://archive-beta.ics.uci.edu/ml/datasets/blood+transfusion+service+center>

I renamed columns for readability as follow:

* last : time in months since last donation
* times : number of blood donations
* unit : accumulated unit of blood donations in cc
* since : time in months since first donation
* donated : whether donated during bus visit in March 2007 (1/0)

Later, I realized this dataset has no distinct features among classification which is not an ideal set to start with.

This dataset can be used for figuring out how many donors may show up in a particular blood drive.

**Dataset and Preparation**

There is no missing data, all column values are integers, no transformation is needed:

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

RangeIndex: 748 entries, 0 to 747

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 last 748 non-null int64

1 times 748 non-null int64

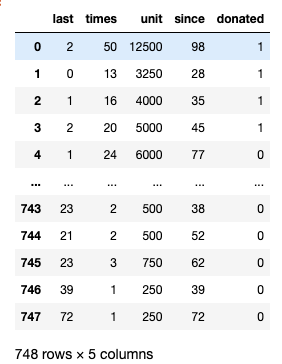
2 unit 748 non-null int64

3 since 748 non-null int64

4 donated 748 non-null int64

dtypes: int64(5)

memory usage: 29.3 KB

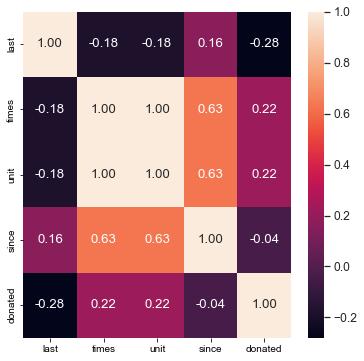


# correlation heatmap

plt.subplots(figsize=(6,6))

sns.set(font\_scale=1.1)

sns.heatmap(blood.corr(), annot=True, fmt='.2f')



As shown above, there is strong correlation between *times* and *unit* as well as between *unit* and *since*. I dropped *unit* and *since* columns:

# drop unit since it is correlated with times

blood = blood.drop(['unit', 'since'], axis=1)

**Baseline Data**

Data is imbalanced, the distribution is 24% donated blood and 76% not:

blood.donated.value\_counts(normalize = True)

0 0.762032

1 0.237968

plt.subplots(figsize=(8,6))

splot = sns.countplot(data=blood, x = 'donated')

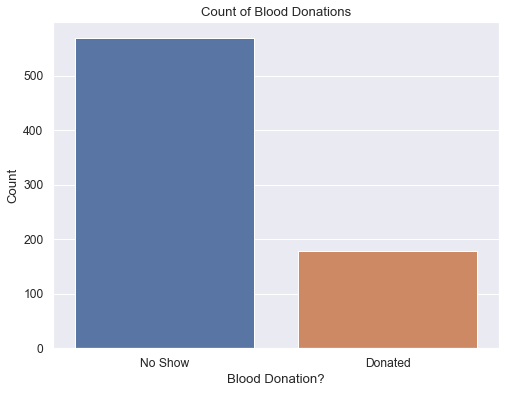
splot.set\_xticklabels(['No Show', 'Donated'])

plt.xlabel('Blood Donation?')

plt.ylabel('Count')

plt.title('Count of Blood Donations')

plt.show()

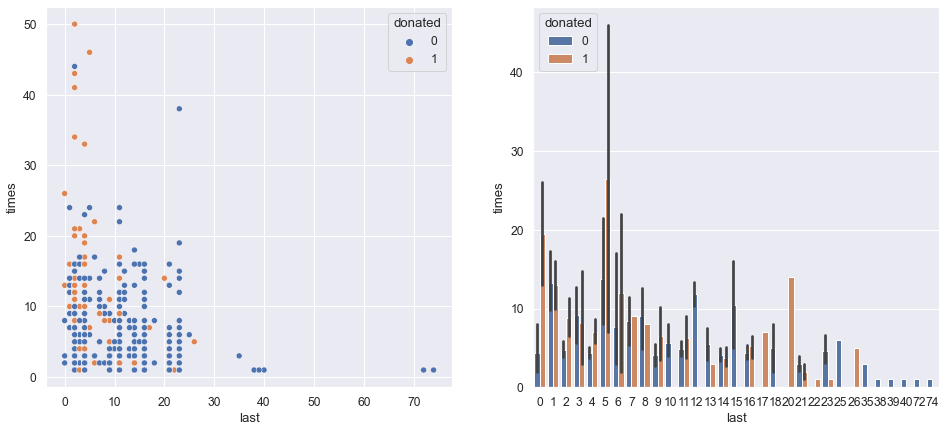


By looking at the scatter and bar plots, the features in dataset are not distinct enough to classify one group from another:

fig, ax = plt.subplots(1, 2, figsize=(16,7))

sns.scatterplot(data=blood, x='last', y='times', hue='donated', ax=ax[0])

sns.barplot(data=blood, x='last', y='times', hue='donated', ax=ax[1])



**Modeling**

There are only 3 columns left in the dataset:

blood.info()

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 last 748 non-null int64

1 times 748 non-null int64

2 donated 748 non-null int64

Split dataset by keeping the same ratio of donors in both train and test ~24%:

# separate features from target and split training and test datasets

X, y = blood.drop('donated', axis = 1), blood.donated

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = 93, stratify = y)

y\_train.value\_counts(normalize = True)

0 0.762923

1 0.237077

y\_test.value\_counts(normalize = True)

0 0.759358

1 0.240642

I created a grid search for finding the best hyper parameters by evaluating n\_neighbors, weights and scoring in GridSearchCV:

params = {'knn\_\_n\_neighbors': range(1, len(y\_test), 2), 'knn\_\_weights':['uniform', 'distance']}

for scorer in ['accuracy', 'precision', 'recall', 'roc\_auc']:

knn\_grid = GridSearchCV(blood\_pipeline, param\_grid = params, scoring=scorer)

knn\_grid.fit(X\_train, y\_train)

best\_k = knn\_grid.best\_params\_

print(scorer + ':', best\_k)

Best runs for each scoring:

accuracy: {'knn\_\_n\_neighbors': 17, 'knn\_\_weights': 'uniform'}

precision: {'knn\_\_n\_neighbors': 71, 'knn\_\_weights': 'uniform'}

recall: {'knn\_\_n\_neighbors': 1, 'knn\_\_weights': 'uniform'}

roc\_auc: {'knn\_\_n\_neighbors': 47, 'knn\_\_weights': 'uniform'}

I created all those models and compared them in ROC Curve Display Plot:

fig, ax = plt.subplots(figsize=(10,8))

RocCurveDisplay.from\_estimator(blood\_pipeline, X\_test, y\_test, pos\_label = 1, ax = ax, label = '5 Neighbors: Default')

RocCurveDisplay.from\_estimator(accuracy\_pipeline, X\_test, y\_test, pos\_label = 1, ax = ax, label = '17 Neighbors: Accuracy')

RocCurveDisplay.from\_estimator(recall\_pipeline, X\_test, y\_test, pos\_label = 1, ax = ax, label = '1 Neighbor: Recall')

RocCurveDisplay.from\_estimator(auc\_pipeline, X\_test, y\_test, pos\_label = 1, ax = ax, label = '47 Neighbors: AUC')

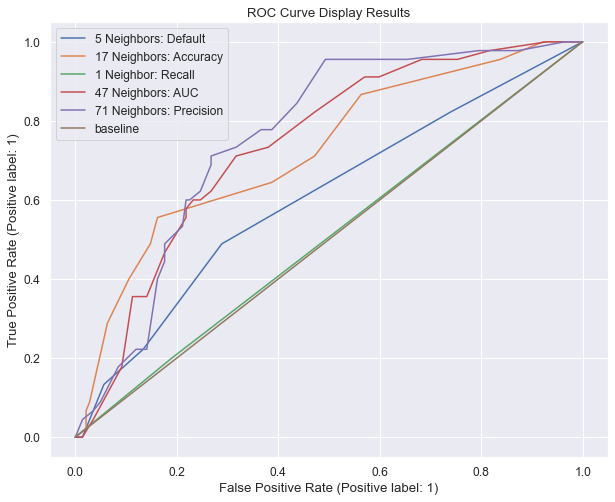
RocCurveDisplay.from\_estimator(precision\_pipeline, X\_test, y\_test, ax = ax, label = '71 Neighbors: Precision')

plt.plot(np.arange(0, 1.1, .1), np.arange(0, 1.1, .1), label = 'baseline')

plt.title('ROC Curve Display Results')

plt.legend()

plt.grid(True)



I decided to go with scoring by the result of ‘roc\_auc’ Area Under Curve outputted as:

# go with roc\_auc

auc\_pipeline = Pipeline([('scale', StandardScaler()),

('knn', KNeighborsClassifier(n\_neighbors = 47, weights = 'uniform'))])

auc\_pipeline.fit(X\_train, y\_train)

auc\_preds = auc\_pipeline.predict(X\_test)

auc\_proba = auc\_pipeline.predict\_proba(X\_test)

I built a dataframe for adjusting the decision boundary:

# auc decision boundary

auc\_db = pd.DataFrame(

{'threshold':thresholds,

'accuracy' :[accuracy\_score(y\_test, np.where(auc\_proba[:, 1] >= t, 1, 0) ) for t in thresholds],

'precision':[precision\_score(y\_test, np.where(auc\_proba[:, 1] >= t, 1, 0),

pos\_label=1, zero\_division=0) for t in thresholds],

'recall' :[recall\_score(y\_test, np.where(auc\_proba[:, 1] >= t, 1, 0),

pos\_label=1, zero\_division=0) for t in thresholds]

})

Visualizing decision thresholds:

plt.subplots(figsize=(10,8))

plt.plot(auc\_db['threshold'], auc\_db['accuracy'], '--o', label = 'accuracy')

plt.plot(auc\_db['threshold'], auc\_db['precision'], '--o', label = 'precision')

plt.plot(auc\_db['threshold'], auc\_db['recall'], '--o', label = 'recall')

plt.axvline(x=0.4, color="black", label = 'intersection')

plt.xticks(thresholds)

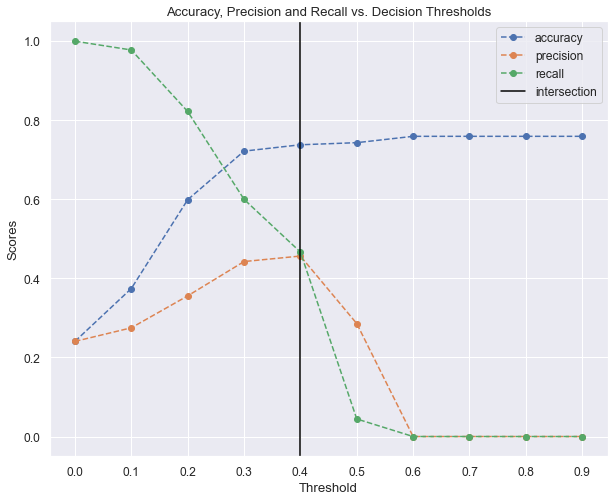
plt.xlabel('Threshold')

plt.ylabel('Scores')

plt.title('Accuracy, Precision and Recall vs. Decision Thresholds')

plt.legend()

plt.grid(True)



# Precision vs Recall

precision, recall, boundaries = precision\_recall\_curve(y\_true = y\_test,

probas\_pred=auc\_proba[:, 1], pos\_label=1)

# plot

plt.subplots(figsize=(10,8))

plt.plot(precision, recall, '--o')

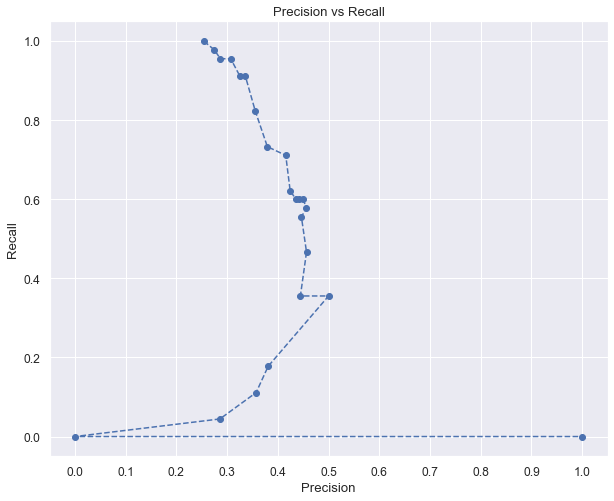
plt.xticks(np.arange(0, 1.1, .1))

plt.grid(True)

plt.ylabel('Recall')

plt.xlabel('Precision')

plt.title('Precision vs Recall')



As *recall* went down *precision* peaked at 0.5 and went down and jumped back at 1.

**Adjusting the decision boundary**

For fine-tuning I picked the intersection point 0.4 as low threshold and 0.6 as the high threshold.

# adjust decision boundary

low\_preds = np.where(auc\_proba[:, 1] > .4, 1, 0)

high\_preds = np.where(auc\_proba[:, 1] > .6, 1, 0)

print(sum((~low\_preds) & y\_test), sum(~auc\_preds & y\_test), sum(~high\_preds & y\_test))

# plot confusion matrices

fig, ax = plt.subplots(1, 3, figsize = (16, 3))

ConfusionMatrixDisplay.from\_predictions(y\_test, low\_preds, display\_labels = ['No Show', 'Donated'], ax = ax[0])

ConfusionMatrixDisplay.from\_predictions(y\_test, auc\_preds, display\_labels = ['No Show', 'Donated'], ax = ax[1])

ConfusionMatrixDisplay.from\_predictions(y\_test, high\_preds, display\_labels = ['No Show', 'Donated'], ax = ax[2])

ax[0].grid(False)

ax[1].grid(False)

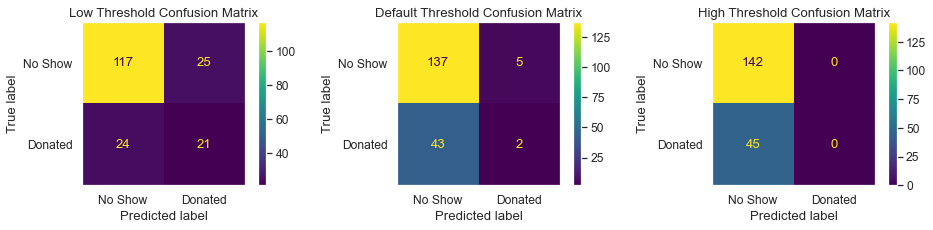
ax[2].grid(False)

ax[0].set\_title('Low Threshold Confusion Matrix')

ax[1].set\_title('Default Threshold Confusion Matrix')

ax[2].set\_title('High Threshold Confusion Matrix')

plt.show()



**Conclusion**

If the goal is to set foot traffic for the blood drive bus, I pick hyperparameters as 'n\_neighbors' = 47 and 'weights' = 'uniform’, in addition to those adjusted the decision boundary at 0.4 which helps as shown on the left chart above. Although, half of predictions are false negative, the number of donors may show up that day within the range of actual number of donors. Again, this dataset turned out to be not an ideal dataset to work with.

 ————— o —————