CS 616 MACHINE LEARNING FINAL PROJECT

HEART DISEASE CORRELATIONS

HEART DISEASE DATASET

- The dataset used contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. Creator(V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.)
- The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).
- This project only utilized the Cleveland database subset and updated values of either 0 or 1 within the target field, thereby adjusting this dataset set to be classified in a binary fashion.

THE BEGINNING…

- import pandas as pd
- import matplotlib.pyplot as plt
- import numpy as np
- # to make this notebook's output stable across runs
- np.random.seed(42)
- #This dataset is of 303 individuals with 14 attributes. The Target value is the presence of heart disease. A 1 indicates heart disease is present.
- #I will attempt to find factors involved in identifying heart disease and look for classifiers that affect the target field.

- heart = pd.read_csv("C:/Users/Owner/Desktop/CS 616 - Machine Learning/Datasets/heart.csv")

VALUE MEANINGS

- Age = age in years
- Sex = (1=male; 0=female)
- CP = Chest Pain Type
- Val 1 = typical angina, Val 2 = atypical angina, Val 3 = Non-anginal pain, Val 4 = asymptomatic
- trestbps = Resting blood pressue (in mm Hg on admission)
- chol = serum cholestoral in mg/dl
- fbs = (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg = resting electrocardiographic results
- Value 0 = normal, Value 1 = have ST-T wave abnormality, Value 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria
- thalach = max heart rate achieved
- exang = exercised induced angina(1=yes; 2=no)
- oldpeak = ST depression induced by exercise relative to rest
- slope = the slope of the peak exercise ST segment
- Value 1 = upsloping, Value 2 = flat, Value 3 = downsloping
- ca = number of major vessels (0-3) colored by flouroscopy
- thal = A blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)
- target = 1 or 0

CHECKING FOR NULL VALUES

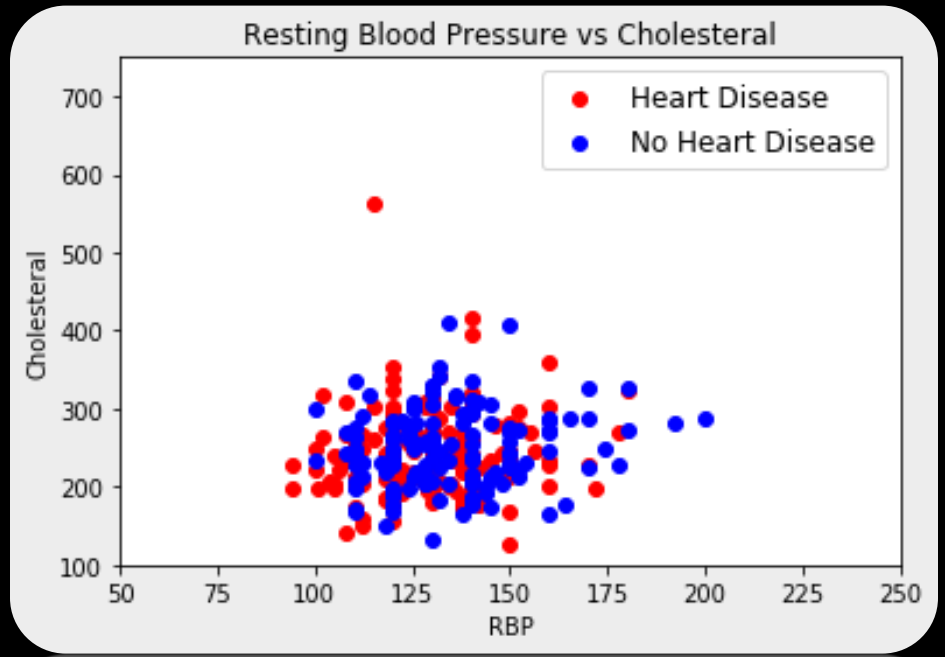
- #Checking for null values
- heart.isnull().sum()
- age 0
- sex 0
- cp 0
- trestbps 0
- chol 0
- fbs 0
- restecg 0
- thalach 0
- exang 0
- oldpeak 0
- slope 0
- ca 0
- thal 0
- target 0
- dtype: int64
- Fortunately we can see there are no null values

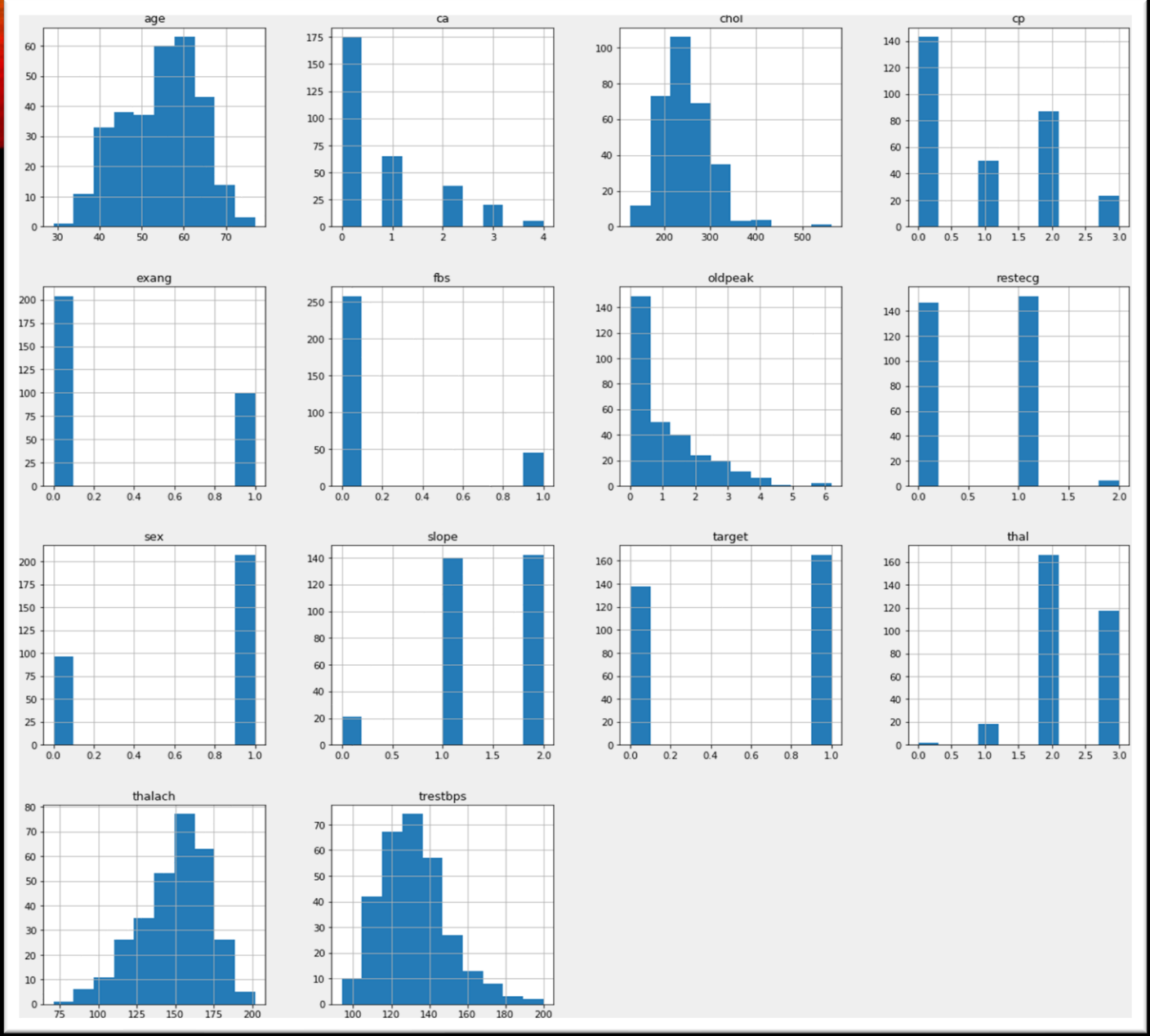
PLOTTING SOME FIELDS

Let us take a look at some commonly believed contributors of heart disease such as cholesterol:

```
#Looking at some commonly believed contributors to heart disease
HD = heart[heart['target']==1.0]
plt.scatter(HD['trestbps'],HD['chol'], color='Red', label='Heart Disease'  )
NHD = heart[heart['target']==0.0]
plt.scatter(NHD['trestbps'],NHD['chol'], color='Blue', label='No Heart Disease'  )
plt.title('Resting Blood Pressure vs Cholesterol')
plt.xlabel('RBP')
plt.ylabel('Cholesterol')
plt.xlim(50,250)
plt.ylim(100,750)
plt.legend(loc="upper right", fontsize=12)
plt.show()
```

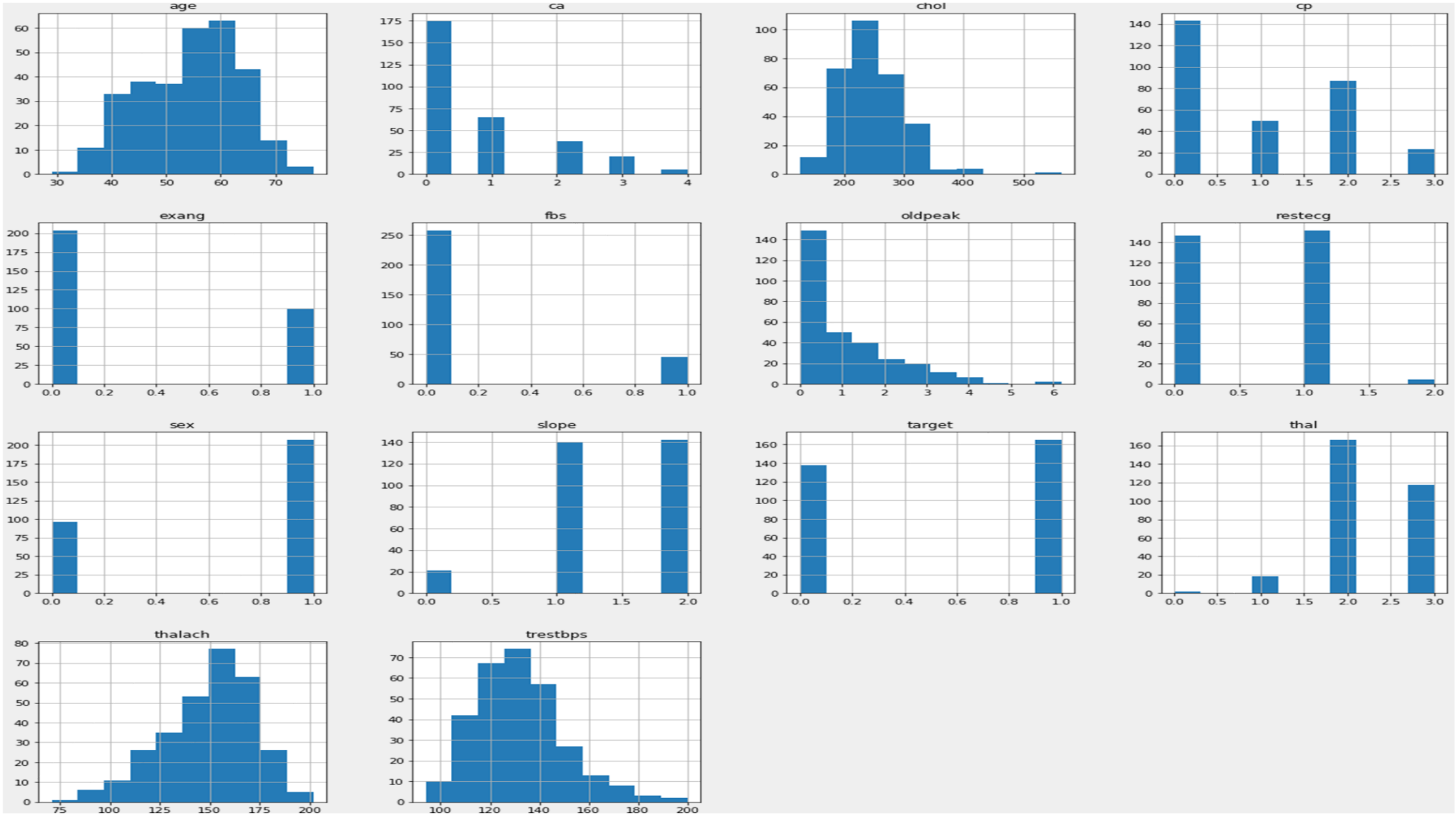
After looking at the graph below we cannot clearly witness a specific contribution of either cholesterol or resting blood pressure to heart disease

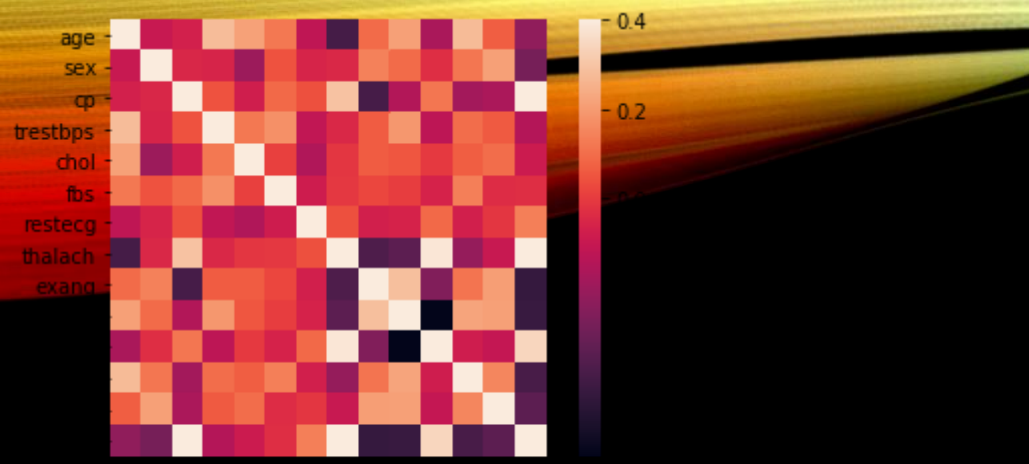


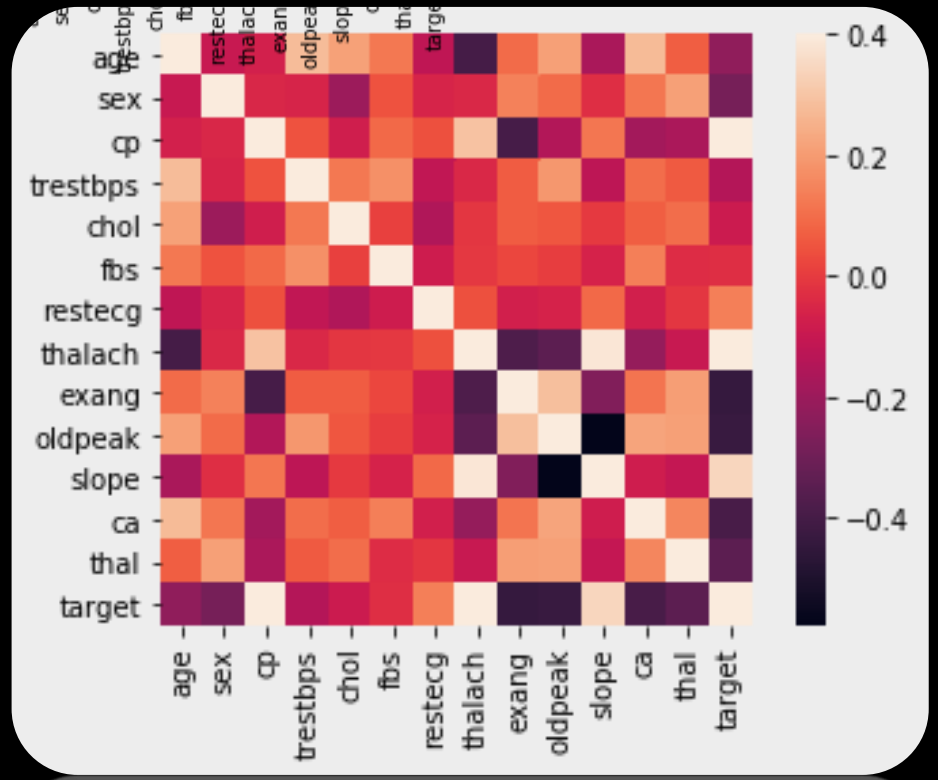


HISTOGRAM

- # Plot histograms of each parameter
- heart.hist(figsize = (20,20))
- plt.show()


age
ca
chol
cp
exang
fbs
oldpeak
restecg
sex
slope
target
thal
thalach
trestbps





HEATMAP CORRELATION

- Let's use a heatmap to check for correlations.
- import seaborn as sns
- sns.heatmap(heart.corr(), vmax = .4, square = True)
- fig = plt.figure(figsize = (60, 60))
- plt.show()
- The most glaring indicator is Slope/Oldpeak.
- #Looks like CP, Exang, and Thalach are also three options to look at it.

DROPPING UNNECESSARY FIELDS

- heart = heart.drop(['age','sex','trestbps','chol','fbs','restecg','thal','oldpeak','slope'],axis=1)
- heart.head()
- First we will be analyzing the 4 fields Chest Pain, Max Heart Rate achieved, Exercise induced angina, and number of major blood vessels colored by fluoroscopy.
- heart.tail()

HEAD

	cp	thalach	exang	ca	target
0	3	150	0	0	1
1	2	187	0	0	1
2	1	172	0	0	1
3	1	178	0	0	1
4	0	163	1	0	1

TAIL

	cp	thalach	exang	ca	target
298	0	123	1	0	0
299	3	132	0	0	0
300	0	141	0	2	0
301	0	115	1	1	0
302	1	174	0	1	0

ADDITIONAL DESCRIPTON OF ANALYZED FIELDS

- CP is Chest Pain:
- -- Value 1: typical angina
  -- Value 2: atypical angina
  -- Value 3: non-anginal pain
  -- Value 4: asymptomatic
- Thalach is maximum heart rate achieved
- Exang is exercise induced angina (1 = Yes, 0 = No)
- CA is number of major vessels (0-3) colored by flouroscopy

SCALING DOWN THE DATA

- from sklearn.preprocessing import MinMaxScaler
- min_max=MinMaxScaler()

- #Scaling heart dataset with Min Max Scaler
- heart_min_max=min_max.fit_transform(heart)

SPLITTING THE TRAIN AND TEST DATA

- #Splitting the train and test, dropping the target field
- from sklearn.model_selection import train_test_split
- train_set, test_set = train_test_split(heart,test_size=0.2,random_state=42)
- train_set.head()
- #Y train is this column only, X train is everyone else.
- Y_train = train_set['target']
- X_train = train_set.drop(['target'],axis=1)
- Y_test = test_set['target']
- X_test = test_set.drop(['target'],axis=1)

CREATING ASSIGNMENT OF TARGET VALUE

- Y_train_1 = (Y_train == 1) # binary array of true/false
- Y_test_1 = (Y_test == 1)

LOGISTIC REGRESSION CLASSIFIER

- #Applying the Logistic Regression classifier
- from sklearn.linear_model import LogisticRegression
- log_reg = LogisticRegression(solver="liblinear", random_state=42)
- log_reg.fit(X_test, Y_test_1)
- LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
- intercept_scaling=1, max_iter=100, multi_class='warn',
- n_jobs=None, penalty='l2', random_state=42, solver='liblinear',
- tol=0.0001, verbose=0, warm_start=False)

RUNNING PREDICTION

- log_reg.predict(X_test)

```
array([False,  True,  True, False,  True,  True, False, False, False,
        True,  True,  True,  True, False,  True,  True,  True, False,
       False, False,  True, False, False, False,  True,  True,  True,
        True, False,  True, False, False, False, False,  True, False,
        True,  True,  True,  True, False,  True,  True,  True,  True,
       False, False,  True, False, False, False, False,  True,  True,
       False, False, False,  True, False, False,  True])

#Checking accuracy of the prediction
from sklearn.metrics import accuracy_score
accuracy_score(Y_test_1,log_reg.predict(X_test))
0.8524590163934426
#Wow, this looks pretty good!
```

```
Out[14]:
array([[ 77,  32],  [ 20, 113]], dtype=int64)
```

CONFUSION MATRIX AND CROSS VALIDATION PREDICTION AND ACCURACY

- from sklearn.model_selection import cross_val_predict
- Y_train_pred = cross_val_predict(log_reg, X_train, Y_train_1, cv=10)
- from sklearn.metrics import confusion_matrix
- confusion_matrix(Y_train_1, Y_train_pred)
- array([[77, 32],
- [20, 113]], dtype=int64)
- from sklearn.model_selection import cross_val_score
- cross_val_score(log_reg, X_train, Y_train_1, cv=10, scoring="accuracy")
- array([0.68 , 0.76 , 0.84 , 0.79166667, 0.83333333,
- 0.75 , 0.75 , 0.91666667, 0.79166667, 0.73913043])

ACCURACY, PRECISION, & RECALL

- #Checking accuracy of the prediction
- from sklearn.metrics import accuracy_score
- accuracy_score(Y_test,log_reg.predict(X_test))
- 0.8524590163934426
- #Checking the precision and recall score
- from sklearn.metrics import precision_score, recall_score
- precision_score(Y_train_1,Y_train_pred)
- 0.7793103448275862
- recall_score(Y_train_1,Y_train_pred)
- 0.849624060150376

HARD VOTING ENSEMBLE

```
#Running an ensemble with Hard voting
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression(solver="liblinear", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=10, random_state=42)
svm_clf = SVC(gamma="auto", random_state=42, probability = True)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')

voting_clf.fit(X_train, Y_train_1)
```

ENSEMBLE HARD VOTING ACCURACY

- from sklearn.metrics import accuracy_score

- for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
- clf.fit(X_train, Y_train_1)
- Y_pred = clf.predict(X_test)
- print(clf.__class__.__name__, accuracy_score(Y_test_1, Y_pred))

- LogisticRegression 0.819672131147541
- RandomForestClassifier 0.7377049180327869
- SVC 0.6885245901639344
- VotingClassifier 0.7868852459016393

SOFT VOTNG ENSEMBLE

```
#Running an ensemble with Soft voting
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression(solver="liblinear", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
svm_clf = SVC(gamma="auto", random_state=42, probability = True)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='soft')

voting_clf.fit(X_train, Y_train_1)
```

ENSEMBLE SOFT VOTING ACCURACY

- from sklearn.metrics import accuracy_score

- for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
- clf.fit(X_train, Y_train_1)
- Y_pred = clf.predict(X_test)
- print(clf.__class__.__name__, accuracy_score(Y_test_1, Y_pred))

- LogisticRegression 0.819672131147541
- RandomForestClassifier 0.7377049180327869
- SVC 0.6885245901639344
- VotingClassifier 0.819672131147541

- So far Logistic Regression is the winner with our first run at .85 or .81 within the ensemble.

NOW WE WILL TRY SLOPE AND OLDPEAK

- #oldpeak = ST depression induced by exercise relative to rest
- What is ST depression you may ask?
- The ST segment represents the isoelectric period when the ventricles are in between depolarization and repolarization.
- A ST depression test that indicates a high probability of **coronary** artery disease is one in which there is substantial ST depression at low work rate associated with typical angina-like pain and a drop in blood pressure. Deeper and more widespread ST depression generally indicates more **severe** or extensive disease.
- #slope = the slope of the peak exercise ST segment
- slope: the slope of the peak exercise ST segment
  -- Value 1: upsloping
  -- Value 2: flat
  -- Value 3: downsloping

STARTING OVER AND DROPPING DIFFERENT FIELDS

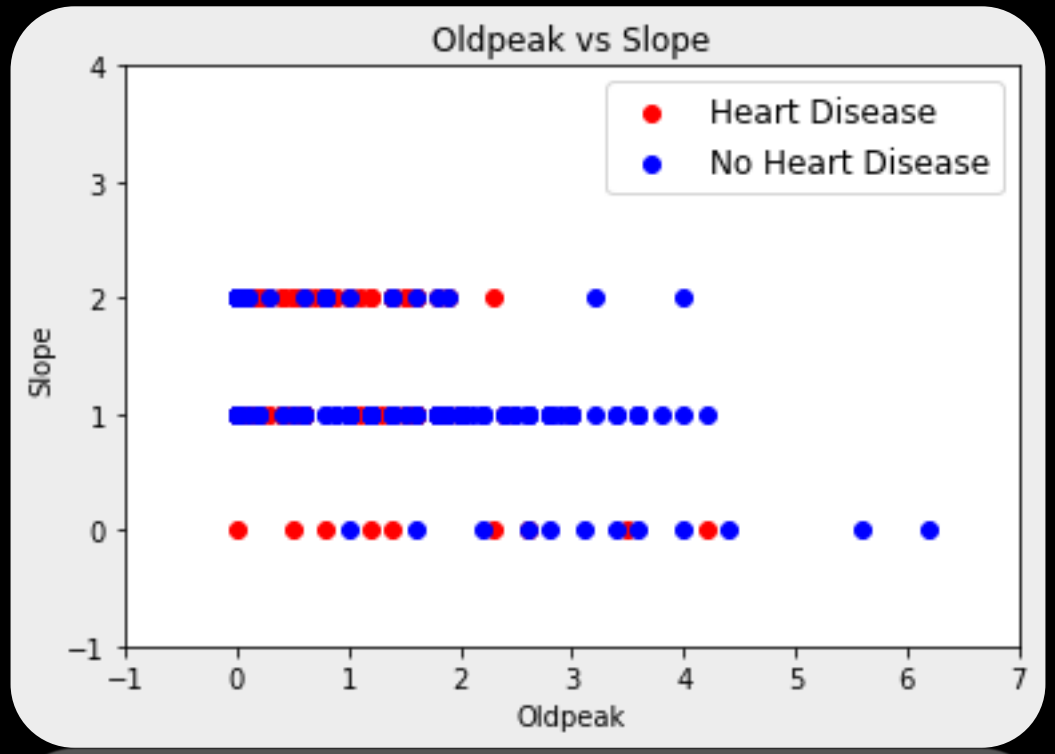
- heart = heart.drop(['age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','ca','thal'],axis=1)
- heart.head()

	oldpeak	slope	target
0	2.3	0	1
1	3.5	0	1
2	1.4	2	1
3	0.8	2	1
4	0.6	2	1

CREATING A NEW SCATTER PLOT FOR A QUICK VIEW

- HD = heart[heart['target']==1.0]
- plt.scatter(HD['oldpeak'],HD['slope'], color='Red', label='Heart Disease')
- NHD = heart[heart['target']==0.0]
- plt.scatter(NHD['oldpeak'],NHD['slope'], color='Blue', label='No Heart Disease')
- plt.title('Oldpeak vs Slope')
- plt.xlabel('Oldpeak')
- plt.ylabel('Slope')
- plt.xlim(-1.0,7.0)
- plt.ylim(-1.0,4.0)
- plt.legend(loc="upper right", fontsize=12)
- plt.show()

HMMM, NOT MUCH HERE TO SEE



PREPPING THE NEW DATA.

- from sklearn.model_selection import train_test_split
- train_set, test_set = train_test_split(heart,test_size=0.2,random_state=42)
- train_set.head()
- #Y train is this column only, X train is everyone else.
- Y_train = train_set['target']
- X_train = train_set.drop(['target'],axis=1)
- Y_test = test_set['target']
- X_test = test_set.drop(['target'],axis=1)
- Y_train_1 = (Y_train == 1) # binary array of true/false
- Y_test_1 = (Y_test == 1)

MIN MAX SCALER AGAIN

- from sklearn.preprocessing import MinMaxScaler
- min_max=MinMaxScaler()

- #Scaling heart dataset with Min Max Scaler
- heart_min_max=min_max.fit_transform(heart)

LOGISTIC REGRESSION CLASSIFER

- #Applying the Logistic Regression classifier
- from sklearn.linear_model import LogisticRegression
- log_reg = LogisticRegression(solver="liblinear", random_state=42)
- log_reg.fit(X_test, Y_test_1)
- log_reg.predict(X_test)
- array([True, True, True, False, True, True, True, False, False,
- False, True, False, True, False, True, True, True, False,
- False, False, True, False, False, True, True, True, True,
- True, False, True, True, True, False, False, False, False,
- False, True, True, True, True, True, True, False, True,
- True, False, True, False, True, True, True, True, True,
- True, False, False, True, False, False, False])

ACCURACY OF PREDICTION

- #Checking accuracy of the prediction
- from sklearn.metrics import accuracy_score
- accuracy_score(Y_test_1,log_reg.predict(X_test))
- 0.7377049180327869
- #This is worse than before…

CONFUSION MATRIX

- from sklearn.model_selection import cross_val_predict
- Y_train_pred = cross_val_predict(log_reg, X_train, Y_train_1, cv=3)
- from sklearn.metrics import confusion_matrix
- confusion_matrix(Y_train_1, Y_train_pred)
- array([[66, 43],
- [27, 106]], dtype=int64)
- #Checking the precision and recall score
- from sklearn.metrics import precision_score, recall_score
- precision_score(Y_train_1,Y_train_pred)
- 0.711409395973154 3
- recall_score(Y_train_1,Y_train_pred)
- 0.7969924812030075

HARD VOTING ENSEMBLE AGAIN

```
#Running an ensemble with Hard voting
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC


log_clf = LogisticRegression(solver="liblinear", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=10, random_state=42)
svm_clf = SVC(gamma="auto", random_state=42, probability = True)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')

voting_clf.fit(X_train, Y_train_1)
```

SURVEY SAYS…

- from sklearn.metrics import accuracy_score

- for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
- clf.fit(X_train, Y_train_1)
- Y_pred = clf.predict(X_test)
- print(clf.__class__.__name__, accuracy_score(Y_test_1, Y_pred))
- LogisticRegression 0.7377049180327869
- RandomForestClassifier 0.7377049180327869
- SVC 0.7704918032786885
- VotingClassifier 0.7704918032786885
- #Worse than our other criteria…

SOFT VOTING ENSEMBLE AGAIN…

```
#Running an ensemble with Soft voting
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression(solver="liblinear", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
svm_clf = SVC(gamma="auto", random_state=42, probability = True)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='soft')

voting_clf.fit(X_train, Y_train_1)
```

SURVEY SAYS…

- from sklearn.metrics import accuracy_score

- for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
- clf.fit(X_train, Y_train_1)
- Y_pred = clf.predict(X_test)
- print(clf.__class__.__name__, accuracy_score(Y_test_1, Y_pred))
- LogisticRegression 0.7377049180327869
- RandomForestClassifier 0.7540983606557377
- SVC 0.7704918032786885
- VotingClassifier 0.7704918032786885

- Definitely not trending in the direction we want.

LETS TRY A DECISION TREE

- from sklearn.tree import DecisionTreeClassifier
- tree_clf = DecisionTreeClassifier(random_state=42)
- tree_clf.fit(X_train, Y_train_1)
- Y_pred_tree = tree_clf.predict(X_test)
- print(accuracy_score(Y_test_1, Y_pred_tree))
- 0.7377049180327869
- #Worse…

BAGGING WITH 500 ATTEMPTS

- from sklearn.ensemble import BaggingClassifier
- bag_clf = BaggingClassifier(
- DecisionTreeClassifier(random_state=42), n_estimators=500,
- max_samples=100, bootstrap=True, n_jobs=-1, random_state=42)
- bag_clf.fit(X_train, Y_train_1)
- Y_pred = bag_clf.predict(X_test)
- print(accuracy_score(Y_test, Y_pred))
- 0.754098360655737
- #Okay so definitely not going the right direction if we just analyze these two fields…

RANDOM FOREST CLASSIFIER

- from sklearn.ensemble import RandomForestClassifier

- rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1, random_state=42)
- rnd_clf.fit(X_train, Y_train_1)
- Y_pred = rnd_clf.predict(X_test)
- print(accuracy_score(Y_test_1, Y_pred))
- 0.754098360655737

OKAY NOW WE WILL TRY ALL OF THE HIGHEST CORRELATIONS TOGETHER

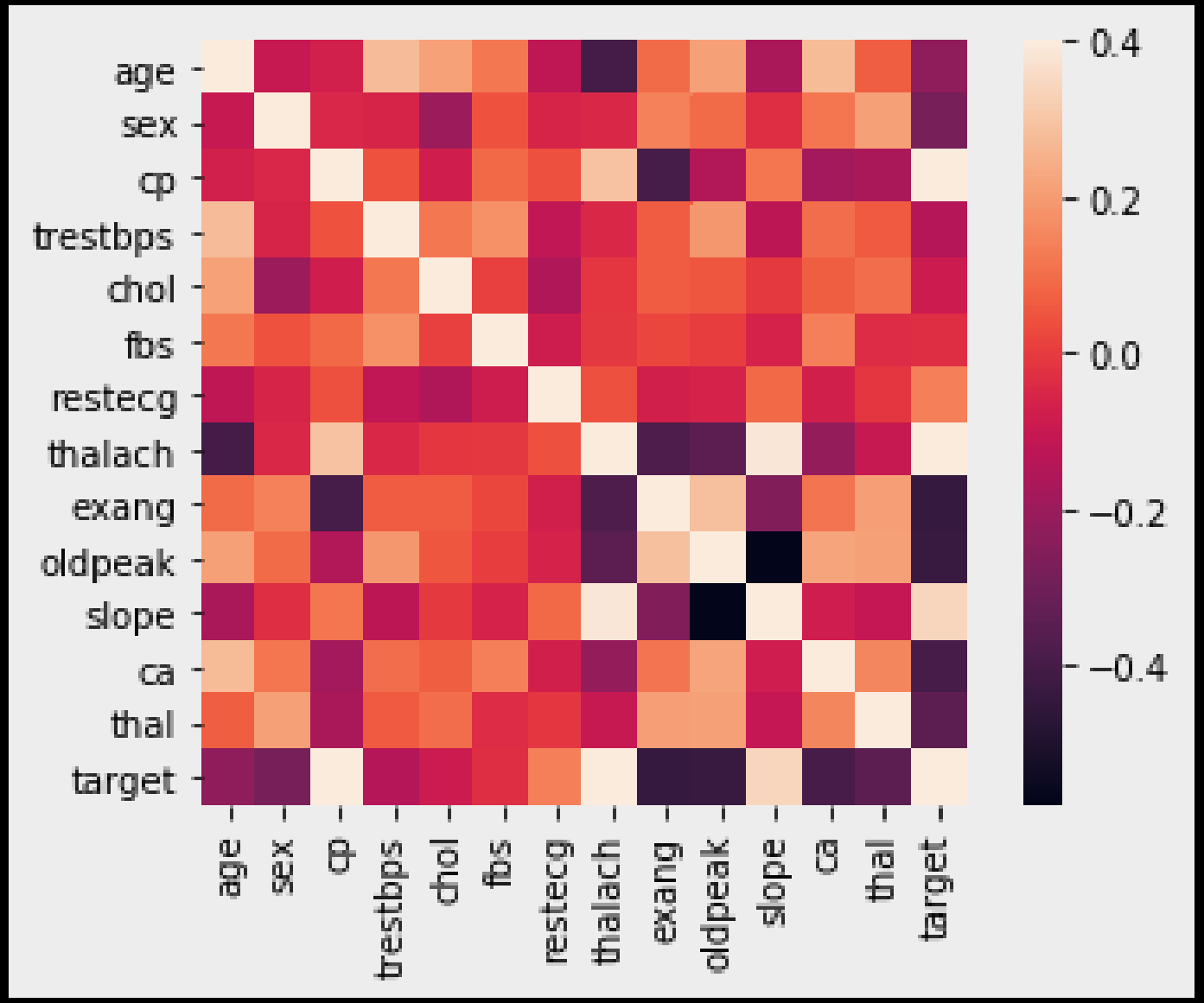
- import pandas as pd
- import matplotlib.pyplot as plt
- import numpy as np
- # to make this notebook's output stable across runs
- np.random.seed(42)
- #This dataset is of 303 individuals with 14 attributes. The Target value is the presence of heart disease. A 1 indicates heart disease is present.
- #I will attempt to find factors involved in identifying heart disease and look for classifiers that affect the target field.

- heart = pd.read_csv("C:/Users/Owner/Desktop/CS 616 - Machine Learning/Datasets/heart.csv")

HEATMAP AGAIN

- import seaborn as sns
- sns.heatmap(heart.corr(), vmax = .4, square = True)
- fig = plt.figure(figsize = (60, 60))
- plt.show()
- We will take CP, Thalach, Exang, Oldpeak, Slope, CA, and Thal and drop the rest.

	cp	thalach	exang	oldpeak		
		slope	ca	thal	target	
• 0	3	150	0			
	2.3	0	0	1		
	1					
• 1	2	187	0			
	3.5	0	0	2		
	1					
• 2	1	172	0			
	1.4	2	0	2		
	1					
• 3	1	178	0			
	0.8	2	0	2		
	1					
• 4	0	163	1			
	0.6	2	0	2		
	1					



MIN MAX AND DATA SPLIT AGAIN

- from sklearn.preprocessing import MinMaxScaler
- min_max=MinMaxScaler()

- #Scaling heart dataset with Min Max Scaler
- heart_min_max=min_max.fit_transform(heart)

- #Splitting the train and test, dropping the target field
- from sklearn.model_selection import train_test_split
- train_set, test_set = train_test_split(heart,test_size=0.2,random_state=42)
- train_set.head()
- #Y train is this column only, X train is everyone else.
- Y_train = train_set['target']
- X_train = train_set.drop(['target'],axis=1)
- Y_test = test_set['target']
- X_test = test_set.drop(['target'],axis=1)

ISOLATE THE TARGET FIELD DEMOGRAPHIC

- Y_train_1 = (Y_train == 1) # binary array of true/false
- Y_test_1 = (Y_test == 1)

LOGISTIC REGRESSION AGAIN

- #Applying the Logistic Regression classifier
- from sklearn.linear_model import LogisticRegression
- log_reg = LogisticRegression(solver="liblinear", random_state=42)
- log_reg.fit(X_test, Y_test_1)
- LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
- intercept_scaling=1, max_iter=100, multi_class='warn',
- n_jobs=None, penalty='l2', random_state=42, solver='liblinear',
- tol=0.0001, verbose=0, warm_start=False)

LOGISTIC REGRESSION OUTPUT

- log_reg.predict(X_test)
- array([False, True, True, False, True, True, True, False, False,
- True, True, True, True, False, True, True, True, False,
- False, False, True, False, False, True, True, True, True,
- True, False, True, False, False, False, False, True, False,
- True, True, True, True, True, True, True, True, True,
- False, True, True, False, False, False, False, True, True,
- False, False, False, True, False, False, False])
- #Checking accuracy of the prediction
- from sklearn.metrics import accuracy_score
- accuracy_score(Y_test_1,log_reg.predict(X_test))
- 0.9016393442622951
- WOW!! Best yet!

CROSS VALIDATION PREDICTION

- from sklearn.model_selection import cross_val_predict
- Y_train_pred = cross_val_predict(log_reg, X_train, Y_train_1, cv=3)
- from sklearn.metrics import confusion_matrix
- confusion_matrix(Y_train_1, Y_train_pred)
- array([[78, 31],
- [19, 114]], dtype=int64)
- from sklearn.model_selection import cross_val_score
- cross_val_score(log_reg, X_train, Y_train_1, cv=10, scoring="accuracy")
- array([0.76 , 0.8 , 0.8 , 0.83333333, 0.875 ,
- 0.79166667, 0.83333333, 0.91666667, 0.79166667, 0.82608696])

PRECISION AND RECALL

- #Checking the precision and recall score
- from sklearn.metrics import precision_score, recall_score
- precision_score(Y_train_1,Y_train_pred)
- 0.7862068965517242
- recall_score(Y_train_1,Y_train_pred)
- 0.8571428571428571

HARD VOTING…LAST TIME I PROMISE

```
#Running an ensemble with Hard voting
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression(solver="liblinear", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=10, random_state=42)
svm_clf = SVC(gamma="auto", random_state=42, probability = True)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')

voting_clf.fit(X_train, Y_train_1)
```

ENSEMBLE WITH HARD VOTING RESULTS

- voting_clf.fit(X_train, Y_train_1)
- from sklearn.metrics import accuracy_score

- for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
- clf.fit(X_train, Y_train_1)
- Y_pred = clf.predict(X_test)
- print(clf.__class__.__name__, accuracy_score(Y_test_1, Y_pred))
- LogisticRegression 0.8852459016393442
- RandomForestClassifier 0.819672131147541
- SVC 0.7704918032786885
- VotingClassifier 0.8688524590163934
- Logistic regression still coming in first with 0.88!!!

ENSEMBLE WITH SOFT VOTING

```
#Running an ensemble with Soft voting
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression(solver="liblinear", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=1000, random_state=42)
svm_clf = SVC(gamma="auto", random_state=42, probability = True)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='soft')

voting_clf.fit(X_train, Y_train_1)
```

ENSEMBLE WITH SOFT VOTING RESULTS

- from sklearn.metrics import accuracy_score

- for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
- clf.fit(X_train, Y_train_1)
- Y_pred = clf.predict(X_test)
- print(clf.__class__.__name__, accuracy_score(Y_test_1, Y_pred))
- LogisticRegression 0.8852459016393442
- RandomForestClassifier 0.819672131147541
- SVC 0.7704918032786885
- VotingClassifier 0.8688524590163934

A QUICK LOOK AT DECISION TREE AND RANDOM FOREST AS WELL

- from sklearn.tree import DecisionTreeClassifier
- tree_clf = DecisionTreeClassifier(random_state=42)
- tree_clf.fit(X_train, Y_train_1)
- Y_pred_tree = tree_clf.predict(X_test)
- print(accuracy_score(Y_test_1, Y_pred_tree))
- 0.8032786885245902

- from sklearn.ensemble import RandomForestClassifier
- rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1, random_state=42)
- rnd_clf.fit(X_train, Y_train_1)
- Y_pred = rnd_clf.predict(X_test)
- print(accuracy_score(Y_test_1, Y_pred))
- 0.8688524590163934

- Both of these results look pretty solid as well!

SUMMARY

- Based on the evidence and some quick "Wikipedia" research, it looks like blood pressure, cholesterol, and heart rate alone will not determine if you have heart disease, but more of a combination of the number of major vessels colored by a fluoroscopy, chest pain, and the results of an ST depression test. Heart disease seems to be more complex than just high cholesterol and blood pressure. Overall I enjoyed this project and learned a bit about the heart.

IN HONOR OF OUR SLIDES THIS TRIMESTER: