HEART DISEASE PREDICTION

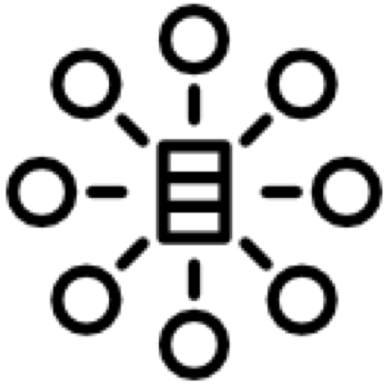
with highest accuracy

DIDEM ISCAN

DSI7 Capstone Project Presentation – January 2019

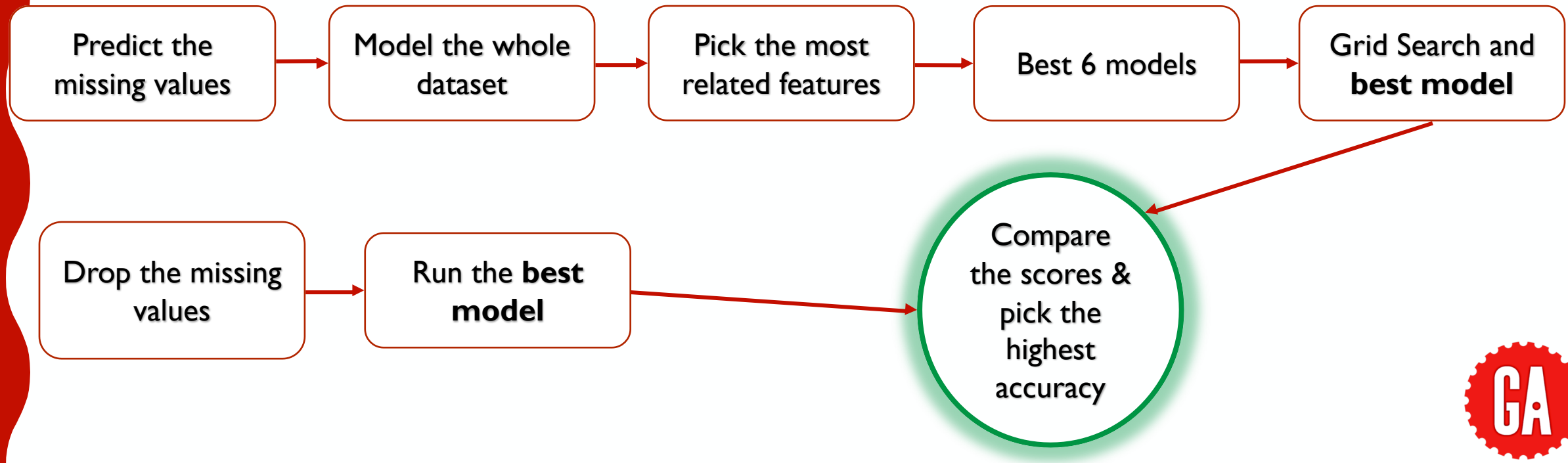
CONTENT

1. Uploading the Data
2. Data Cleaning & EDA
3. Modelling
   - Models
   - Improving the Accuracy
     - Picking the most related features
   - Best model selection
     - Grid Search for best 5 model
   - Dropped vs Predicted Missing Value Comparison
4. Conclusion
5. Next Step & Difficulties



AIM OF THE PROJECT

- Predict the heart disease with the highest accuracy
  - *early prediction means early intervention, and early intervention can save lives.*



HYPOTHESIS

- Missing Value Prediction will increase the disease prediction accuracy

- Picking the most related features will increase the disease prediction accuracy

DATA DESCRIPTIONS

- 4 different datasets
- 16 features
  - 2 different types of variables
    - Continuous: suject id, age, trestpbs, chol, thalach, oldpeak
    - Categorical: sex, cp, fbs, restecg, exang, slop, ca, thal, pred_attribute, origin

Column Name	Description
subject_id	Id number of the subject
Age	Age in years
Sex	Female = 0, Male = 1
CP	Chest pain type
Value 1	Typical angina
Value 2	Atypical angina
Value 3	Non-anginal pain
Value 4	Asymptomatic
Trestbps	Resting blood pressure (in mm Hg on admission to the hospital)
Chol	Serum cholesterol in mg/dl
FBS	(fasting blood sugar > 120 mg/dl) 1 = True, 0 = False
Restecg	Resting electrocardiographic results
Value 0	Normal
Value 1	Having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
Value 2	Showing probable or definite left ventricular hypertrophy by Estes' criteria
Thalach	Maximum heart rate achieved
Exang	Exercise induced angina 1 = Yes, 0 = No
Oldpeak	ST depression induced by exercise relative to rest
Slop	The slope of the peak exercise ST segment
Value 1	Upsloping
Value 2	Flat
Value 3	Downsloping
CA	Number of major vessels (0-3) coloured by fluoroscopy
Thal	Thallium heart scan 3 = normal (no cold spots) 6 = fixed defect (cold spots during rest and exercise) 7 = reversible defect (when cold spots only appear during exercise)
Pred_attribute	The predicted attribute, diagnosis of heart disease (angiographic disease status)
Value 0	< 50% diameter narrowing
Value 1	> 50% diameter narrowing (in any major vessel: attributes 59 through 68 are vessels)
Origin	The origin place of the subject

DATA CLEANING & EDA

	subject_id	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop	ca	thal	pred_attribute	origin
0	0	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0	Cleveland
1	1	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2	Cleveland
2	2	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1	Cleveland
3	3	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0	Cleveland
4	4	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0	Cleveland

Shape: (920,16)

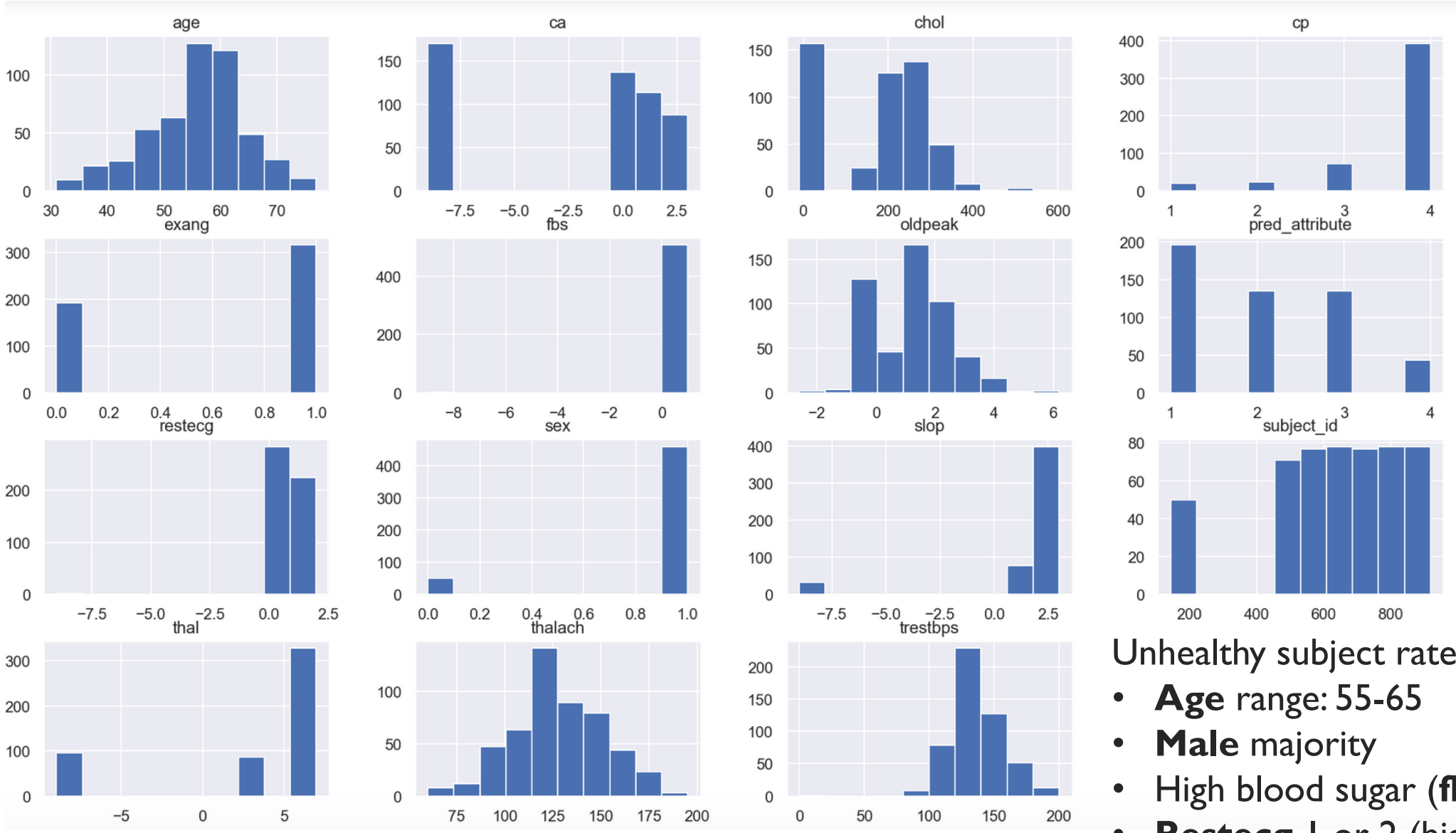
```
data.isnull().sum()
```

```
subject_id          0
age                 0
sex                 0
cp                  0
trestbps           58
chol                7
fbs                82
restecg             1
thalach            54
exang              54
oldpeak            62
slop              119
ca                320
thal              220
pred_attribute      0
origin              0
```

- Many columns contained "?", they all updated to NaN
- Updated the data types into floats
- Predicted the missing values

pred_attributes value count

```
0.0    0.446739
1.0    0.213043
3.0    0.146739
2.0    0.146739
4.0    0.046739
```



Unhealthy subject rates

- **Age** range: 55-65
- **Male** majority
- High blood sugar (**fbs**)
- **Restecg** 1 or 2 (high – shows abnormalities)
  - Electrocardiographic result

PREDICTING MISSING VALUES

Approach I

- Predicting the missing values by using the whole data frame

Approach II

- Predicting the missing values by splitting the data frame into four different categories.
  - Female Healthy
  - Female Unhealthy
  - Male Healthy
  - Male Unhealthy

APPROACH II

1. Split the whole dataset into 4 categories
2. For each data frame
- Compare the scores & predict the missing values
  - Start from smallest number of missing valued feature
  - Two different methods for categorical & continuous variables
    - Pick the best model by comparing with the baseline

subject_id	0
age	0
sex	0
cp	0
trestbps	58
chol	7
fbs	82
restecg	1
thalach	54
exang	54
oldpeak	62
slop	119
ca	320
thal	220
pred_attribute	0
origin	0

`df_female_0.isnull().sum()`

subject_id	0
age	0
sex	0
cp	0
trestbps	1
chol	0
fbs	0
restecg	0
thalach	1
exang	1
oldpeak	1
slop	2
ca	3
thal	3
pred_attribute	0
origin	0

`df_female_1.isnull().sum()`

subject_id	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	7
restecg	0
thalach	0
exang	0
oldpeak	0
slop	4
ca	13
thal	6
pred_attribute	0
origin	0

`df_male_0.isnull().sum()`

subject_id	0
age	0
sex	0
cp	0
trestbps	18
chol	1
fbs	8
restecg	0
thalach	18
exang	18
oldpeak	20
slop	34
ca	57
thal	48
pred_attribute	0
origin	0

`df_male_1.isnull().sum()`

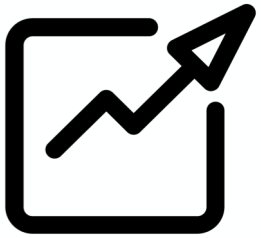
subject_id	0
age	0
sex	0
cp	0
trestbps	39
chol	6
fbs	67
restecg	1
thalach	35
exang	35
oldpeak	41
slop	79
ca	247
thal	163
pred_attribute	0
origin	0

PREDICTING MISSING VALUES

categorical values

Classification Model

- Grid Search
- Compare the scores with the baseline
- Accuracy > Baseline
  - Choose the model & predict
- Accuracy < Baseline
  - Choose the baseline

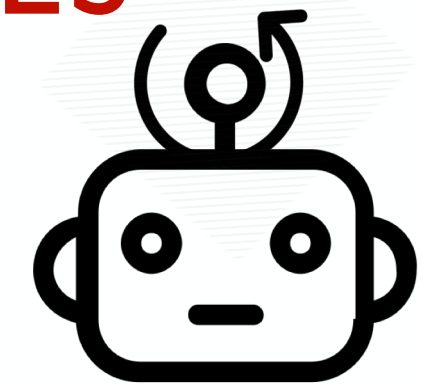


```
#classifier
model_dtc = DecisionTreeClassifier()
model_rfc = RandomForestClassifier()
model_bgc = BaggingClassifier()
```

```
#classifier
params_dtc = {'criterion':['gini', 'entropy'], 'max_depth':np.arange(1,11), 'max_features':range(1,X.shape[1]+1),
              'min_samples_split':[5,10,15,20,25,30,40,50]}
params_rfc = {'criterion': ['gini', 'entropy'], 'n_estimators':np.arange(10,100,10),
              #'max_samples': np.linspace(0.8, 1.0, 3),
              'bootstrap':['True','False'],
              'max_features':['auto', 'sqrt', 'log2'], 'max_depth' : [1, 5, 10, 15, 20, 25, 30],
              'min_samples_leaf' : [1, 2, 4, 6, 8, 10]}
params_bgc = {'base_estimator':[None, Perceptron(),KNeighborsClassifier(),SVC()],
              'n_estimators':np.arange(10,100,10), 'max_samples': np.linspace(0.8, 1.0, 3),
              'max_features':range(int(3/4.*X.shape[1]), X.shape[1]+1), 'bootstrap': ['True','False']}
```

PREDICTING MISSING VALUES

continuous values



Regression Model

- Grid Search
- Compare the scores with the baseline
- Accuracy > 0
  - Choose the model & predict
- Accuracy < 0
  - Random imputation
    - randomly picking values from the y

```
#regressor
model_dtr = DecisionTreeRegressor()
model_rfr = RandomForestRegressor()
model_bgr = BaggingRegressor()
```

```
#regressor
params_dtr = {'max_depth': list(range(1,11))+[None],'max_leaf_nodes': [5, 6, 7, 8, 9, 10],
              'min_samples_leaf': [1, 2, 3, 4], 'min_samples_split':[5,10,15,20,25,30,40,50]}
params_rfr = {'n_estimators': np.arange(10,100,10), 'max_depth': np.linspace(10,10,10),
              'max_features': ['auto','sqrt'], 'bootstrap':['True','False']}
params_bgr = {'base_estimator':[None, DecisionTreeRegressor(),KNeighborsRegressor(),SVR()],
              'n_estimators':np.arange(10,100,10), 'max_samples':np.linspace(0.8, 1.0, 3),
              'max_features':range(int(3/4.*X.shape[1]), X.shape[1]+1), 'bootstrap':['True','False']}
```

AFTER PREDICTING THE MISSING VALUES

```
df.isnull().sum()
```

subject_id	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slop	0
ca	0
thal	0
pred_attribute	0
origin	0

Before	pred_attribute
subject_id	-0.045676
age	0.280461
sex	0.258224
cp	0.405022
trestbps	0.165424
chol	-0.098056
fbs	0.109248
restecg	0.061876
thalach	-0.381984
exang	0.382178
oldpeak	0.469537
slop	0.371813
ca	0.106134
thal	0.211016
pred_attribute	1.000000

Approach I

After	pred_attribute
subject_id	-0.045676
age	0.280461
sex	0.258224
cp	0.405022
trestbps	0.161561
chol	-0.101723
fbs	0.103100
restecg	0.061939
thalach	-0.372177
exang	0.374724
oldpeak	0.454812
slop	0.359766
ca	-0.044599
thal	0.227749
pred_attribute	1.000000

Approach II

After	pred_attribute
subject_id	0.628725
age	0.280461
sex	0.258224
cp	0.405022
trestbps	0.162215
chol	-0.099033
fbs	0.103100
restecg	0.061939
thalach	-0.375818
exang	0.404480
oldpeak	0.477799
slop	0.362316
ca	0.140647
thal	0.270088
pred_attribute	1.000000

MODELLING

1. Define X & y
   - Dummy , standardize, train & test split

	Training	Test	Total
Healthy	351.0	158.0	509.0
Sick	293.0	118.0	411.0

   - Y: pred_attribute (y>0 unhealthy y = 1, y =0 healthy)

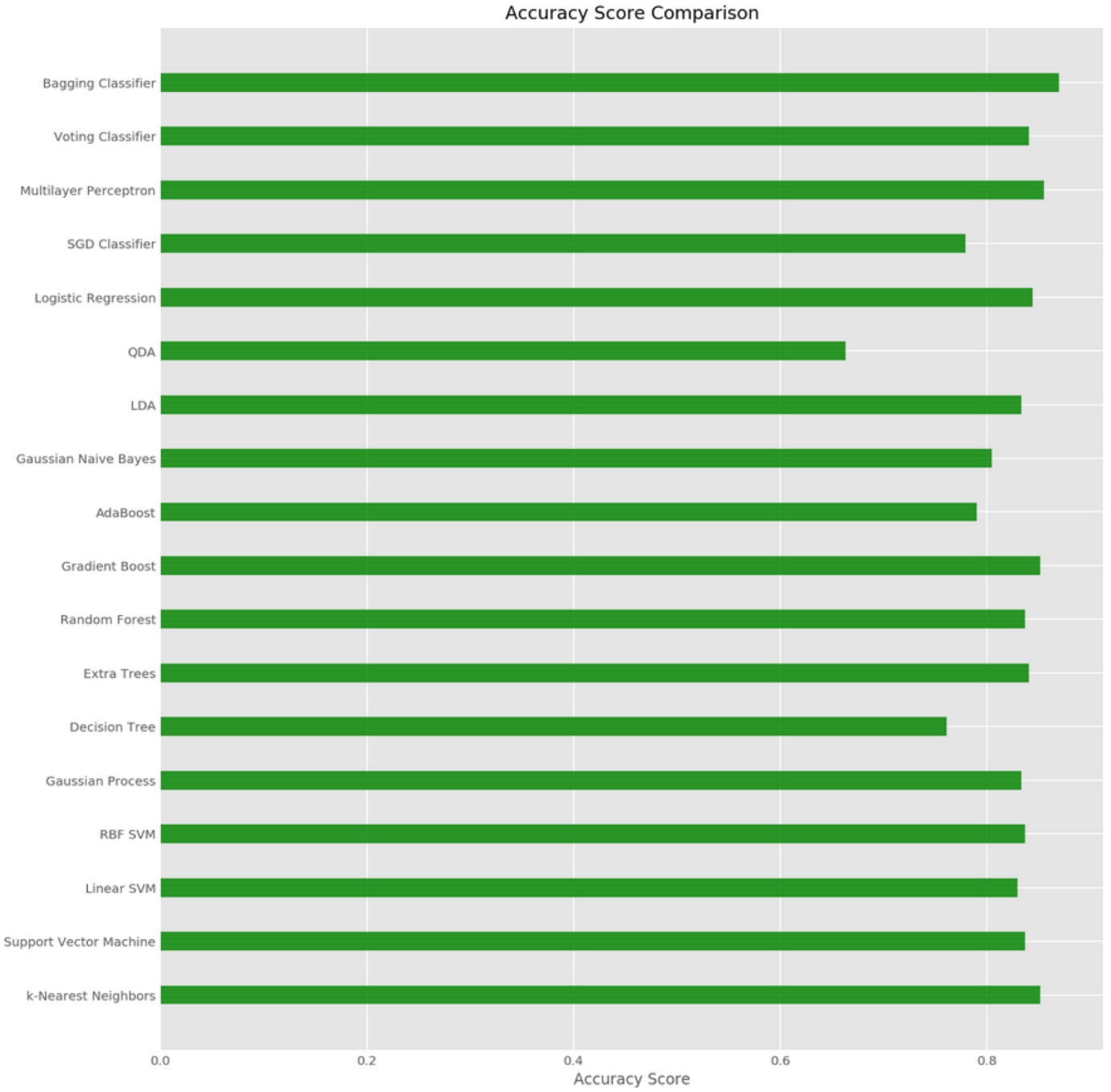
```
y.pred_attribute.value_counts(normalize=True)
print('Baseline:')
print(y.pred_attribute.value_counts(normalize=True))
```

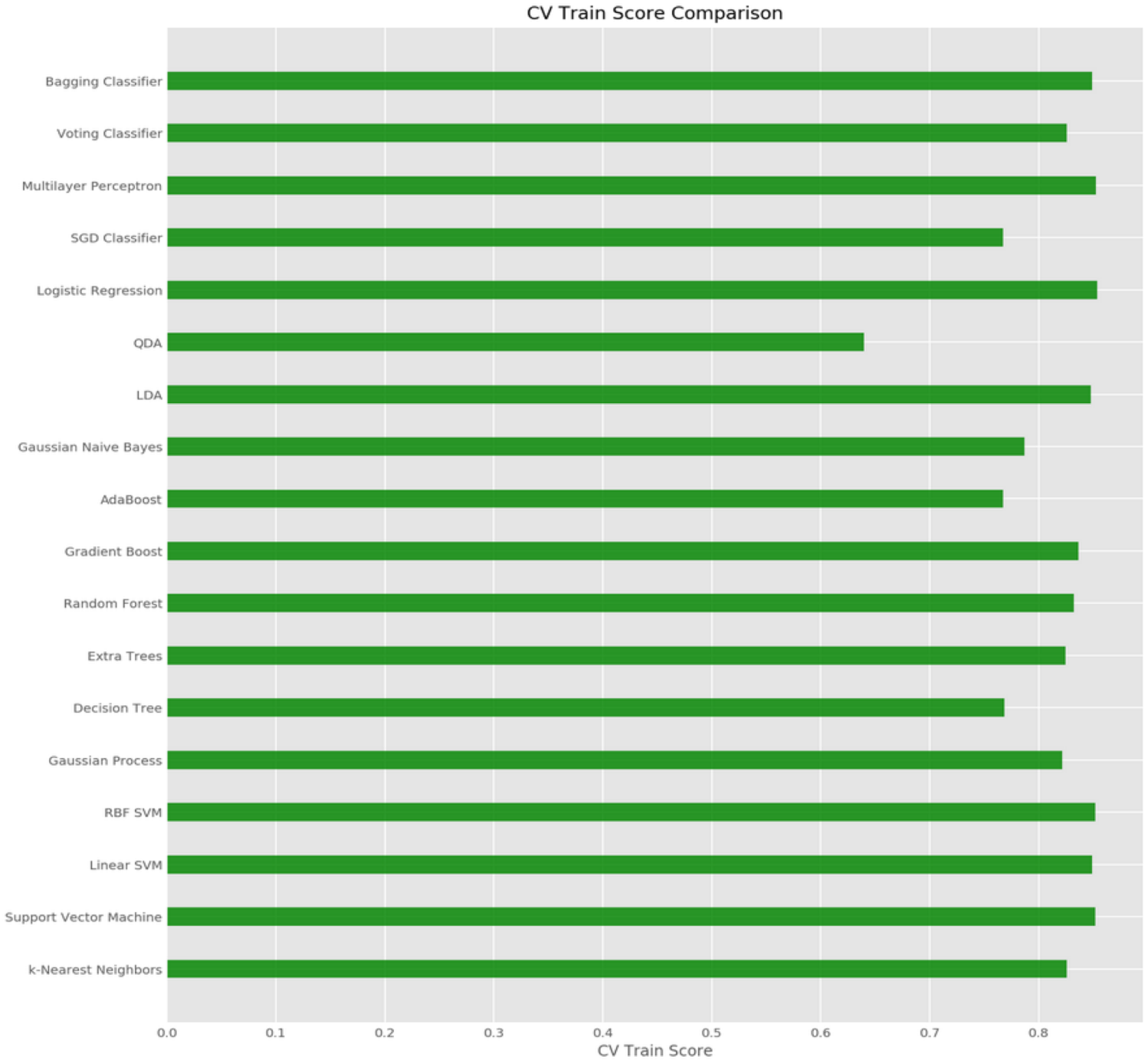
```
Baseline:
1.0    0.553261
0.0    0.446739
```

2. Model Comparison
   - Score Comparison
     - CV Train & Test Scores
     - Train & Test Scores (checking the overfitting)
     - Accuracy Scores
3. Selecting Most Related Columns
   - Effects of modelling with the most related columns
4. Choosing the best model
   - Grid Search the best 6 models
   - Score Comparison
     - CV Train Scores
     - ROC, AUC, Recall, Precision, Scores

	CV Train Score	CV Test Score	Training Data Accuracy Score	Testing Data Accuracy Score	Train-Test Difference
Logistic Regression	0.853777	0.822751	0.863354	0.844203	0.019151
Multilayer Perceptron	0.852311	0.837169	0.874224	0.855072	0.019151
Support Vector Machine	0.852251	0.833862	0.920807	0.836957	0.083851
RBF SVM	0.852251	0.833862	0.920807	0.836957	0.083851
Bagging Classifier	0.849246	0.848016	0.881988	0.869565	0.012422
Linear SVM	0.849102	0.808333	0.861801	0.829710	0.032091
LDA	0.847672	0.844444	0.850932	0.833333	0.017598
Gradient Boost	0.836710	0.818783	0.954969	0.851449	0.103520
Random Forest	0.832059	0.815212	0.993789	0.836957	0.156832
k-Nearest Neighbors	0.825929	0.818783	0.903727	0.851449	0.052277
Voting Classifier	0.825893	0.833730	0.973602	0.836957	0.136646
Extra Trees	0.824319	0.818254	1.000000	0.840580	0.159420
Gaussian Process	0.821338	0.800794	0.990683	0.833333	0.157350
Gaussian Naive Bayes	0.787169	0.797751	0.777950	0.804348	-0.026398
Decision Tree	0.768563	0.753439	1.000000	0.760870	0.239130
SGD Classifier	0.767108	0.793254	0.812112	0.833333	-0.021222
AdaBoost	0.767001	0.760979	1.000000	0.789855	0.210145
QDA	0.639636	0.673016	0.729814	0.663043	0.066770

	AccuracyScore
Bagging Classifier	0.869565
Multilayer Perceptron	0.855072
Gradient Boost	0.851449
k-Nearest Neighbors	0.851449
Logistic Regression	0.844203
Extra Trees	0.840580
Voting Classifier	0.840580
Random Forest	0.836957
RBF SVM	0.836957
Support Vector Machine	0.836957
Gaussian Process	0.833333
LDA	0.833333
Linear SVM	0.829710
Gaussian Naive Bayes	0.804348
AdaBoost	0.789855
SGD Classifier	0.778986
Decision Tree	0.760870
QDA	0.663043





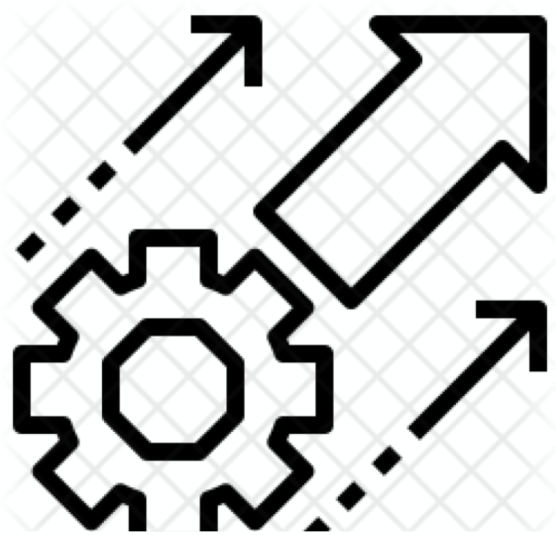
Models with highest score:

- Logistic Regression
- Multilayer Perceptron
- RBF SVM
- Gradient Boost
- Support Vector Machine
- Bagging Classifier

IMPROVING THE ACCURACY SCORE

MOST RELATED COLUMNS

- Find the most related columns
  - Check the correlation
  - Check the best features of models from grid search
    - **oldpeak, exang, slop, cp, thalach, chol**
- Compare the new scores



	pred_attribute
pred_attribute	1.000000
subject_id	0.751197
cp	0.471712
exang	0.436162
slop	0.425163
oldpeak	0.395369
thal	0.354417
sex	0.307284
age	0.282700
ca	0.200564
trestbps	0.112061
fbs	0.094688
restecg	0.047835
chol	-0.187931
thalach	-0.376215

Random Forest

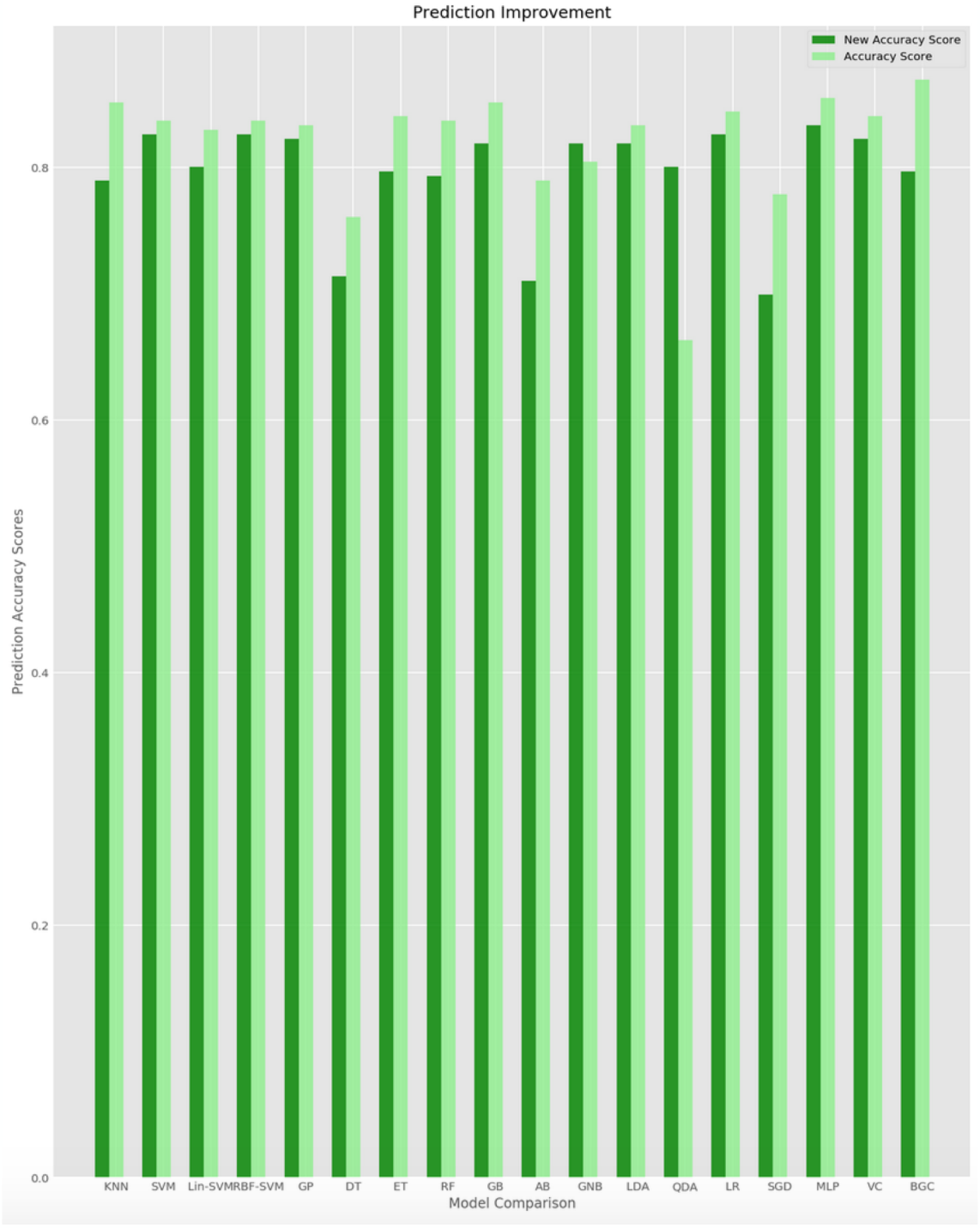
RFC: cp_4.0	0.101243
exang_1.0	0.078534
thal_7.0	0.077146
thalach	0.075852
exang_0.0	0.071418
age	0.070932
chol	0.067766
oldpeak	0.064508
trestbps	0.061591
slop_2.0	0.058658
origin_Switzerland	0.034196
sex_1.0	0.032333
cp_2.0	0.028247
ca_0.0	0.021872
cp_3.0	0.020544
origin_Hungarian	0.018657
thal_3.0	0.017278
slop_1.0	0.014249
restecg_0.0	0.011834
restecg_1.0	0.010760
ca_1.0	0.010463
restecg_2.0	0.009702
origin_LongBeach	0.009456
fbs_1.0	0.007832
fbs_0.0	0.007696
ca_2.0	0.007552
thal_6.0	0.005524
slop_3.0	0.002718
ca_3.0	0.001439
ca_9.0	0.000000

Decision Tree

DTC: cp_2.0	0.257541
exang_1.0	0.220917
origin_Switzerland	0.116456
thal_7.0	0.061155
oldpeak	0.055936
ca_0.0	0.040977
thalach	0.034873
cp_4.0	0.034837
trestbps	0.026532
chol	0.024417
thal_3.0	0.022487
thal_6.0	0.021182
cp_3.0	0.018506
age	0.017476
sex_1.0	0.012468
origin_Hungarian	0.009059
slop_1.0	0.007628
restecg_2.0	0.007580
ca_9.0	0.005622
origin_LongBeach	0.002485
restecg_1.0	0.001167
slop_2.0	0.000699
restecg_0.0	0.000000
fbs_1.0	0.000000
fbs_0.0	0.000000
slop_3.0	0.000000
ca_1.0	0.000000
ca_2.0	0.000000
ca_3.0	0.000000
exang_0.0	0.000000

- The highest importance is the most related.
- **oldpeak, exang, slop, cp, thalach, chol** values are the ones that are most related to y (pred_attribute).

	NewAccuracyScore	AccuracyScore	Difference
Multilayer Perceptron	0.833333	0.855072	-0.021739
RBF SVM	0.826087	0.836957	-0.010870
Support Vector Machine	0.826087	0.836957	-0.010870
Logistic Regression	0.826087	0.844203	-0.018116
Gaussian Process	0.822464	0.833333	-0.010870
Voting Classifier	0.822464	0.840580	-0.018116
Gradient Boost	0.818841	0.851449	-0.032609
Gaussian Naive Bayes	0.818841	0.804348	0.014493
LDA	0.818841	0.833333	-0.014493
Linear SVM	0.800725	0.829710	-0.028986
QDA	0.800725	0.663043	0.137681
Bagging Classifier	0.797101	0.869565	-0.072464
Extra Trees	0.797101	0.840580	-0.043478
Random Forest	0.793478	0.836957	-0.043478
k-Nearest Neighbors	0.789855	0.851449	-0.061594
Decision Tree	0.713768	0.760870	-0.047101
AdaBoost	0.710145	0.789855	-0.079710
SGD Classifier	0.699275	0.778986	-0.079710



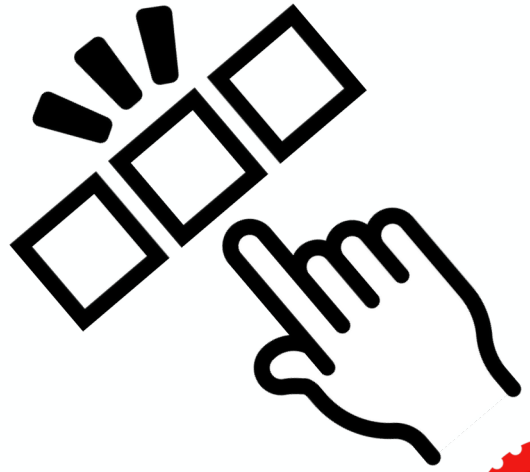
	CV Train Score
Logistic Regression	0.853777
Multilayer Perceptron	0.852311
Support Vector Machine	0.852251
RBF SVM	0.852251
Bagging Classifier	0.849246
Linear SVM	0.849102
LDA	0.847672
Gradient Boost	0.836710
Random Forest	0.832059
k-Nearest Neighbors	0.825929
Voting Classifier	0.825893
Extra Trees	0.824319
Gaussian Process	0.821338
Gaussian Naive Bayes	0.787169
Decision Tree	0.768563
SGD Classifier	0.767108
AdaBoost	0.767001
QDA	0.639636

	AccuracyScore
Bagging Classifier	0.869565
Multilayer Perceptron	0.855072
Gradient Boost	0.851449
k-Nearest Neighbors	0.851449
Logistic Regression	0.844203
Extra Trees	0.840580
Voting Classifier	0.840580
Random Forest	0.836957
RBF SVM	0.836957
Support Vector Machine	0.836957
Gaussian Process	0.833333
LDA	0.833333
Linear SVM	0.829710
Gaussian Naive Bayes	0.804348
AdaBoost	0.789855
SGD Classifier	0.778986
Decision Tree	0.760870
QDA	0.663043

- Choosing the related columns decreased the scores.
- The correlation between columns are not very high, thus when we choose the most related columns it doesn't give us a big difference.
- Next step
  - Grid search on the best models.
  - **Logistic Regression, Multilayer Perceptron, RBF SVM, Gradient Boost, Support Vector Machine, Bagging Classifier**

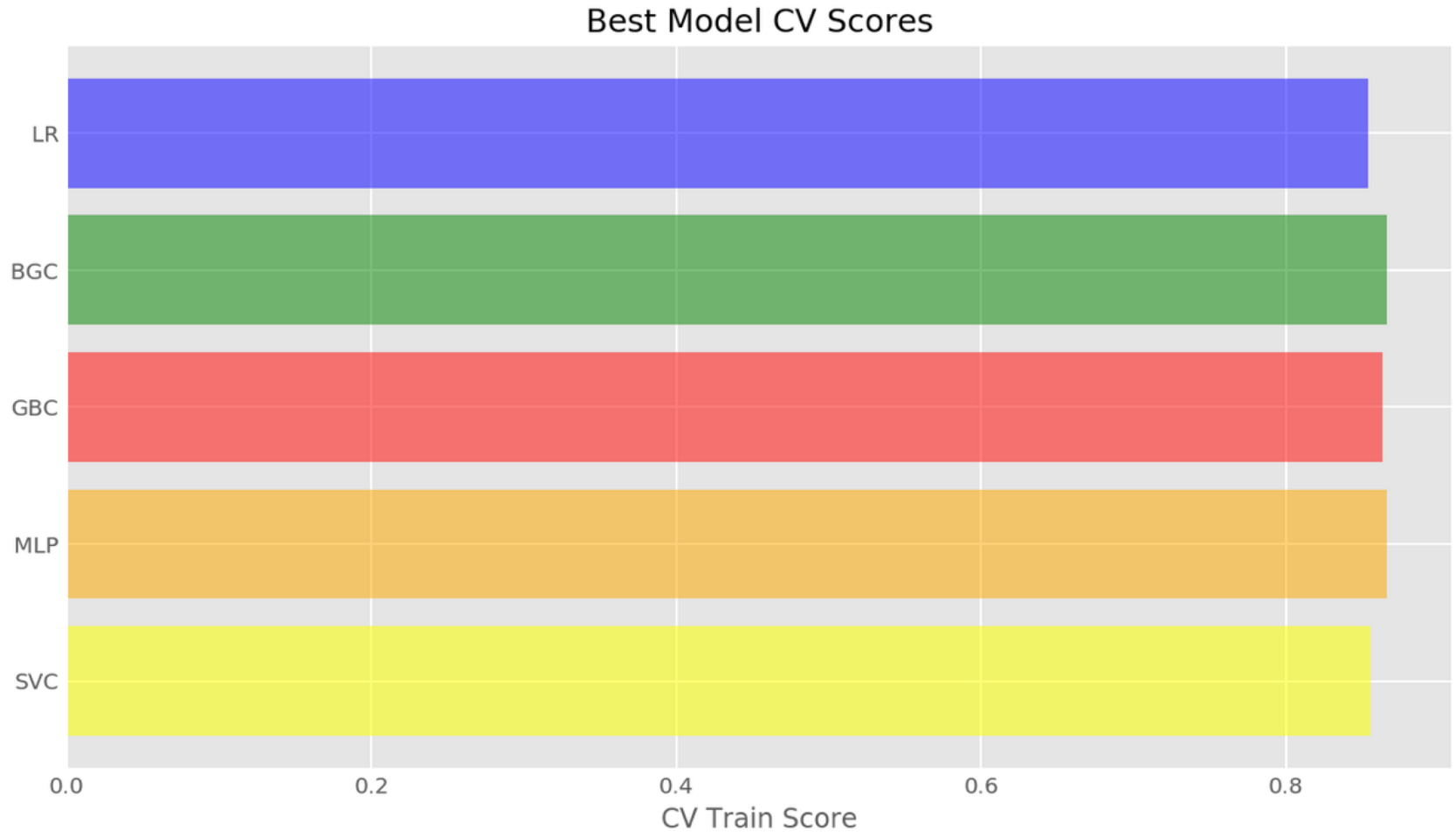
BEST MODEL SELECTION

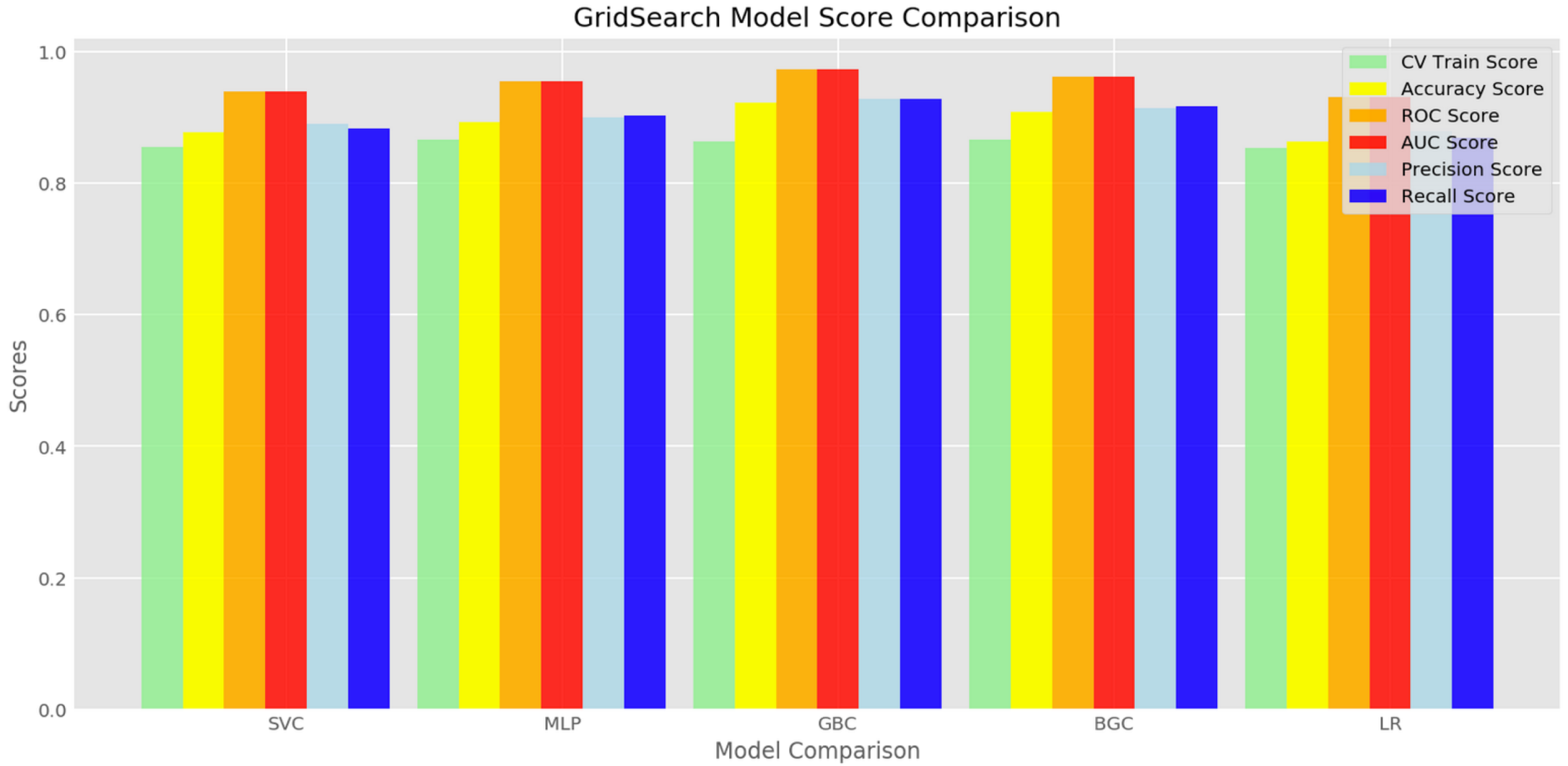
- Choose the best model by using Grid Search
  - Logistic Regression, Multilayer Perceptron, RBF SVM, Gradient Boost, Support Vector Machine, Bagging Classifier
- Grid Search
  - X & y
    - Dummy – standardize – train & test split
  - Grid Search
- Score Comparison
  - Train & Test Scores
  - Accuracy, Precision, Recall, AUC, ROC Scores

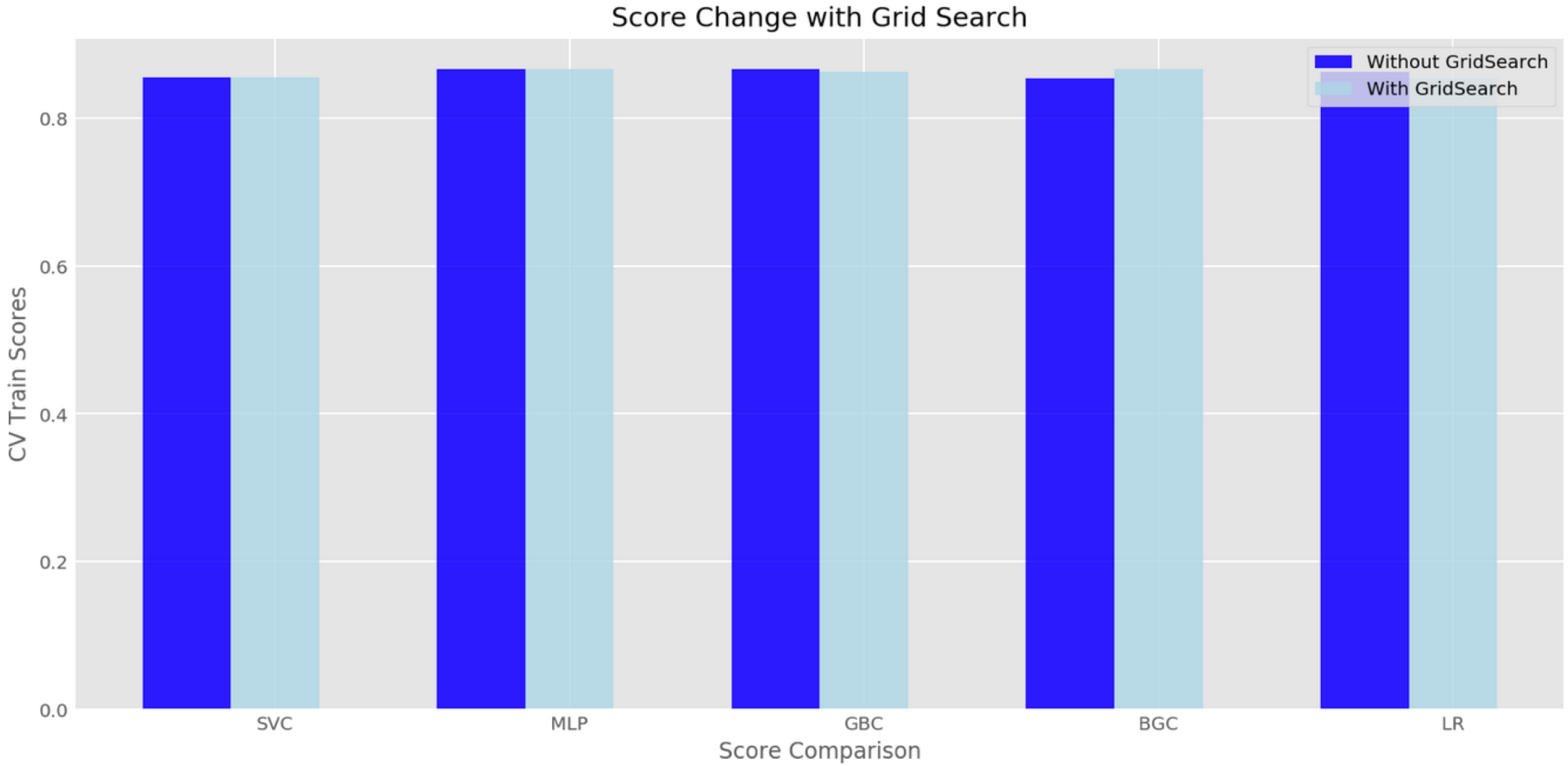


	CV Train Score	CV Test Score	Training Data Accuracy Score	Testing Data Accuracy Score	Train-Test Difference
MLP	0.866241	0.844180	0.892857	0.851449	0.041408
BGC	0.866217	0.833730	0.908385	0.869565	0.038820
GBC	0.863200	0.847751	0.922360	0.844203	0.078157
SVC	0.855376	0.848148	0.877329	0.855072	0.022257
LR	0.853777	0.822751	0.863354	0.844203	0.019151

	CV Train Score	AccuracyScore	ROCScore	AUCScore	PrecisionScore	RecallScore
SVC	0.855376	0.877329	0.939568	0.939568	0.890805	0.883191
MLP	0.866241	0.892857	0.955583	0.955583	0.900568	0.903134
GBC	0.863200	0.922360	0.973027	0.973027	0.928775	0.928775
BGC	0.866217	0.908385	0.962516	0.962516	0.914773	0.917379
LR	0.853777	0.863354	0.931984	0.931984	0.878963	0.868946


Best Model CV Scores
LR
BGC
GBC
MLP
SVC
0.0
0.2
0.4
0.6
0.8
CV Train Score


GridSearch Model Score Comparison
CV Train Score
Accuracy Score
ROC Score
AUC Score
Precision Score
Recall Score
Scores
1.0
0.8
0.6
0.4
0.2
0.0
SVC
MLP
GBC
BGC
LR
Model Comparison



The graph shows us, by using the Grid Search the scores increased or stayed the same.

- After all the analysis and modelling, **MLP, GBC, BGC** have the highest CV Train Scores. While they have similar scores I checked their Accuracy, Precision and Recall Scores as well, and as a result I concluded that **GBC** model gives us the best heart disease prediction. But while GBC model may overfit the data, I decided to pick **BGC** as my best model.
- Next section I used the best model on a different data frame where I didn't predict the missing values and dropped them. This is showing me if predicting missing values increase or decrease the prediction accuracy.
- Additionally **BGC**'s false positive score is 0.04 (diagnosing the subject as healthy but the subject is diseased).

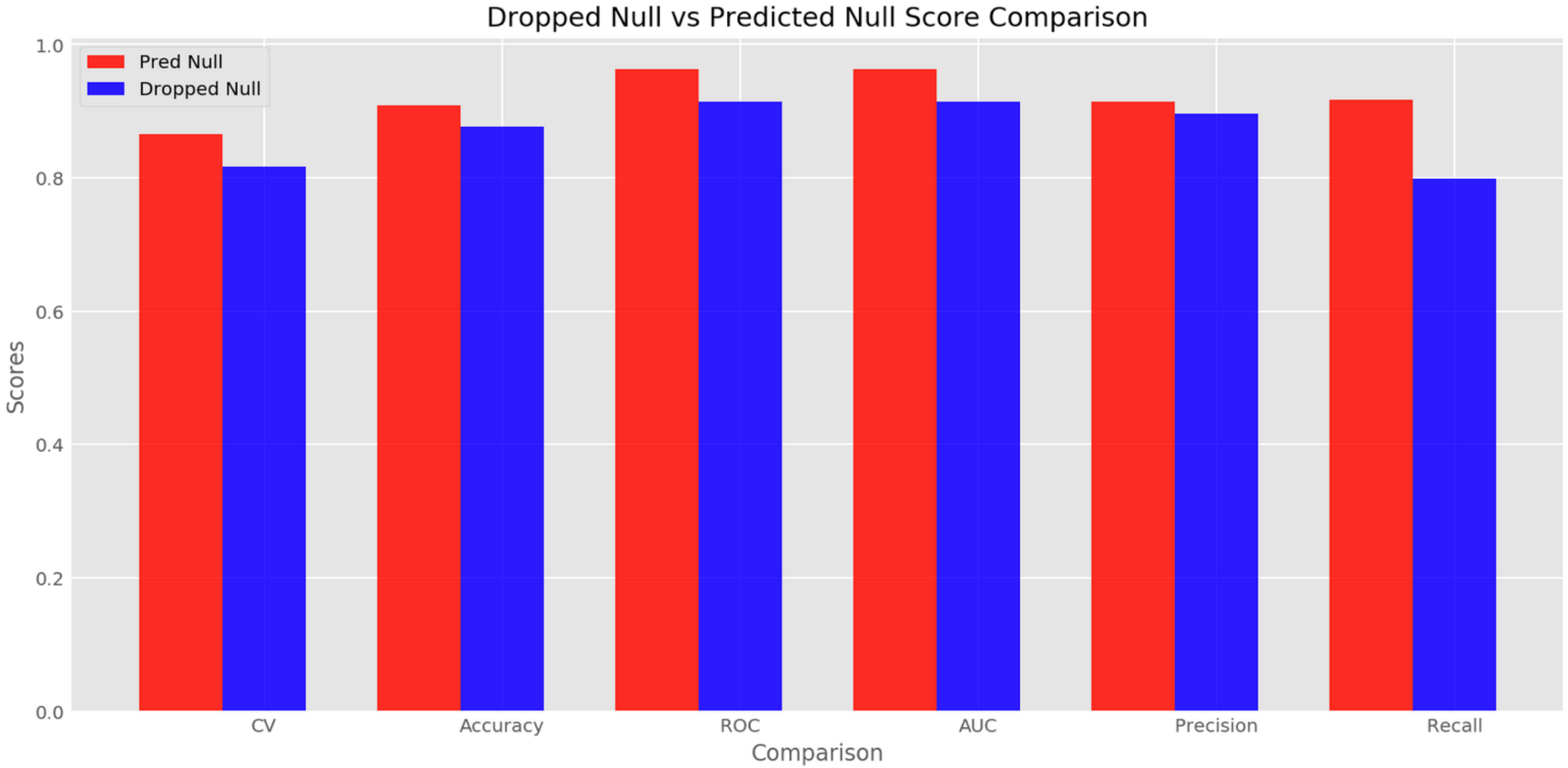
COMPARISON
DROPPED VS. PREDICTED MISSING VALUE

Dropped

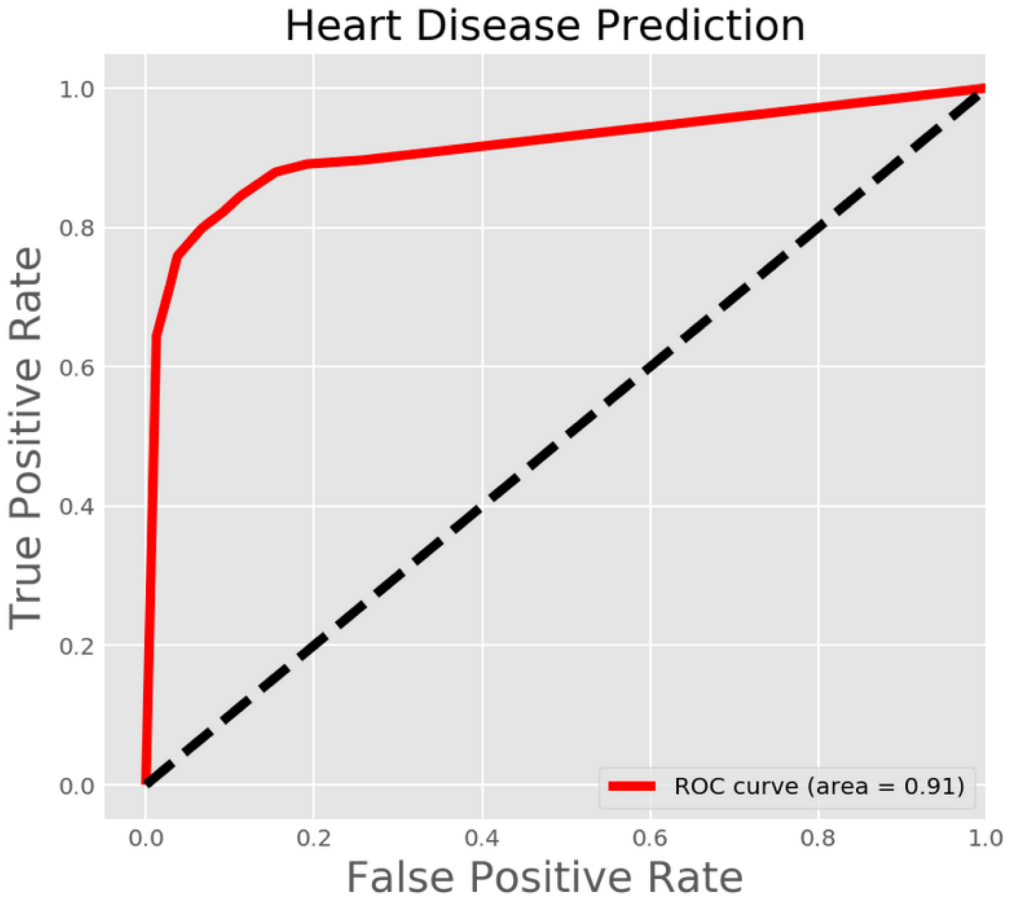
- Shape (592,16)
- Baseline: 0.58
- Accuracy Score: 0.87
- CV Train Score: 0.81
- Recall Score: 0.79
- False Positive: 0.03

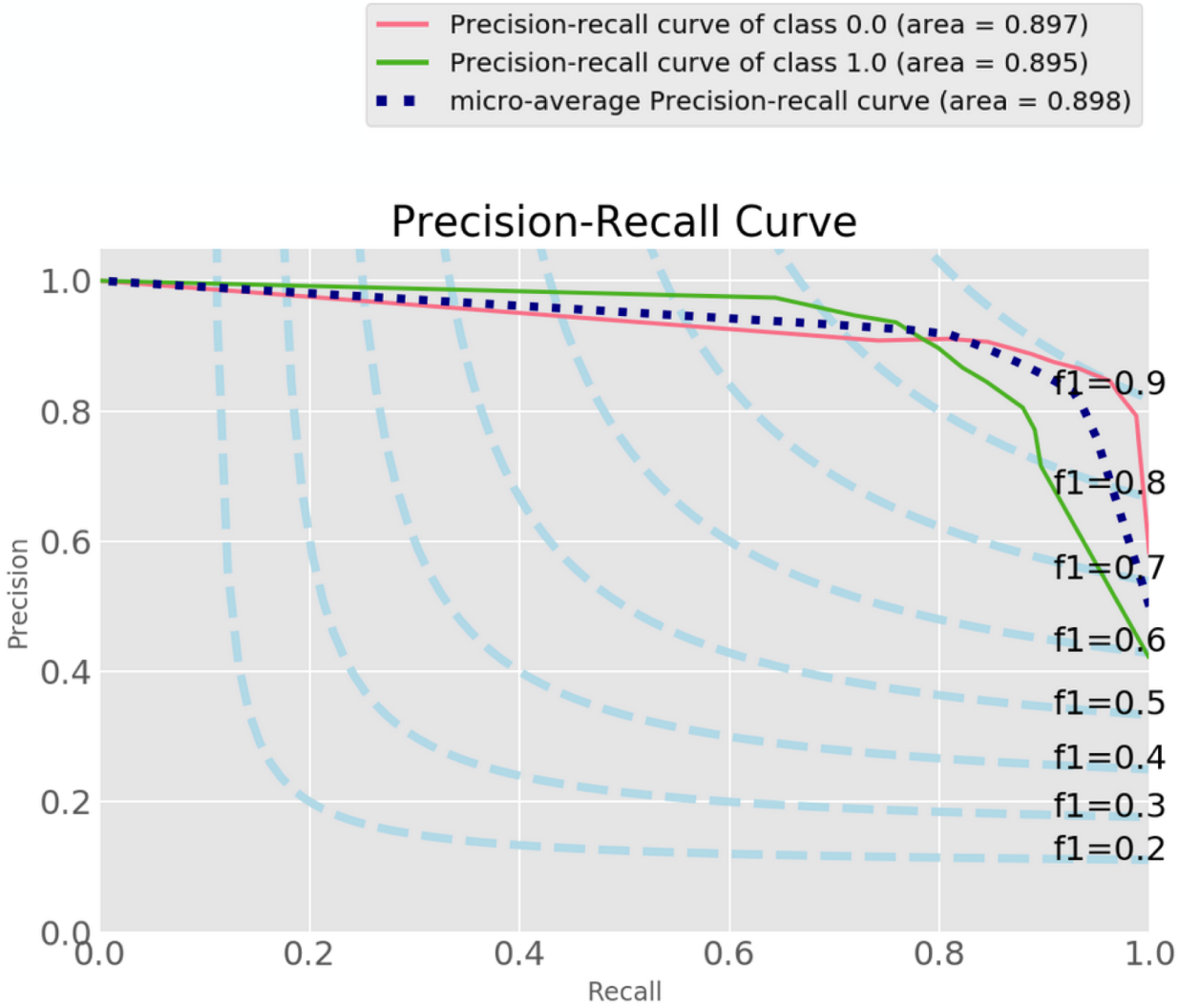
Predicted

- Shape (920,16)
- Baseline: 0.55
- Accuracy Score: 0.90
- CV Train Score: 0.86
- Recall Score: 0.91
- False Positive: 0.04

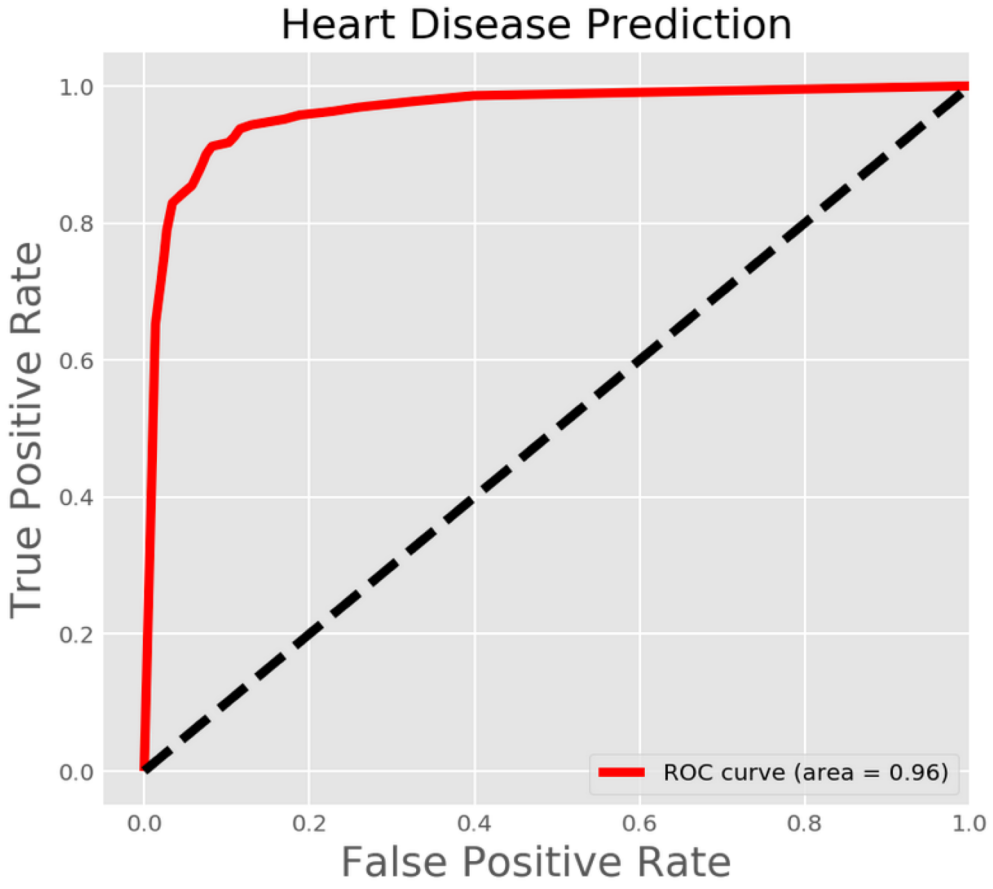

Dropped Null vs Predicted Null Score Comparison
Pred Null
Dropped Null
1.0
0.8
0.6
0.4
0.2
0.0
Scores
CV
Accuracy
ROC
AUC
Precision
Recall
Comparison

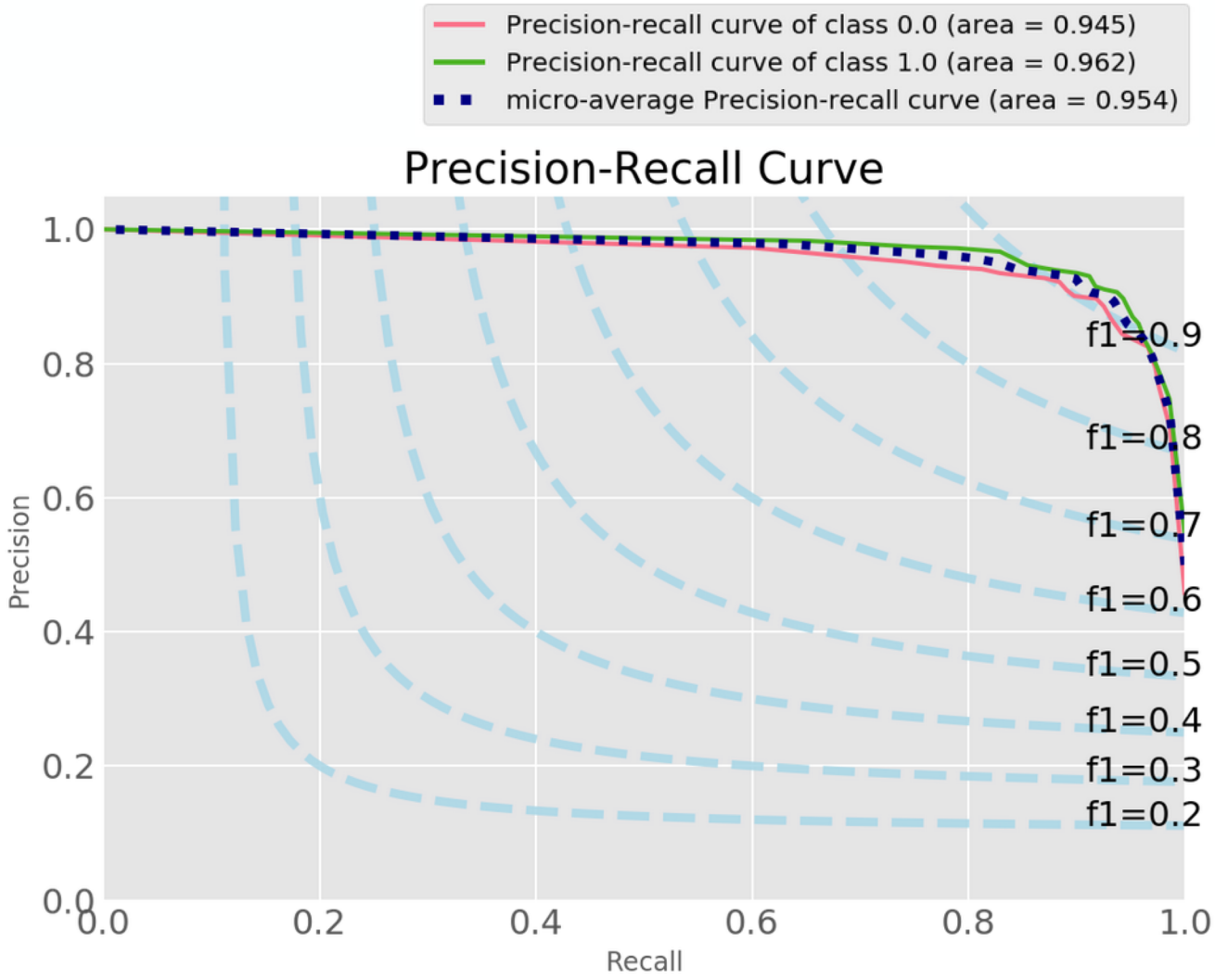
DROPPED MISSING VALUES





PREDICTED MISSING VALUES





CONCLUSION

	CV_TrainScore	AccuracyScore	ROCScore	AUCScore	PrecisionScore	RecallScore
PredictedNull	0.866217	0.908385	0.962516	0.962516	0.914773	0.917379
DroppedNull	0.816456	0.876812	0.913853	0.913853	0.896774	0.798851

- All in all by the analysis we can see that predicting the missing values slightly increases the accuracy of prediction. Additionally, using different modelling for the dataset will help us see different methods and increase the accuracy of our analysis at the end.
- This project helps us to predict a heart disease on a subject with **90%** accuracy (by predicting the missing values), which can help doctors to diagnose heart diseases faster. This will increase the time efficiency on the diagnosis and continue with the further testing faster to cure the diseases.

NEXT STEPS

- Collect more data & work on larger dataset
  - Expect increase in prediction accuracy scores
  - Predict more specific diseases
  - Give patients recommendations
    - Which tests to take
    - How to continue treatment
    - Which exercises they can do

THANK YOU!

"Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world."

Atul Butte