Heart Disease Stroke data science project

In this ML project, I have used my skills to acquire a dataset from kaggle and (https://www.kaggle.com/ronitf/heart-disease-uci/data) and I will use ML techniques to predict wheather the person might get a stroke or not

Here we will be using 4 algorithms

1. K Nearest Neighbours Classifier
2. Random Forest Classifier
3. Decision Tree Classifier
4. Logistic Regression

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000

Here we just did dft.describe and dft.info to find out the most basic info about the dataset such as the mean std

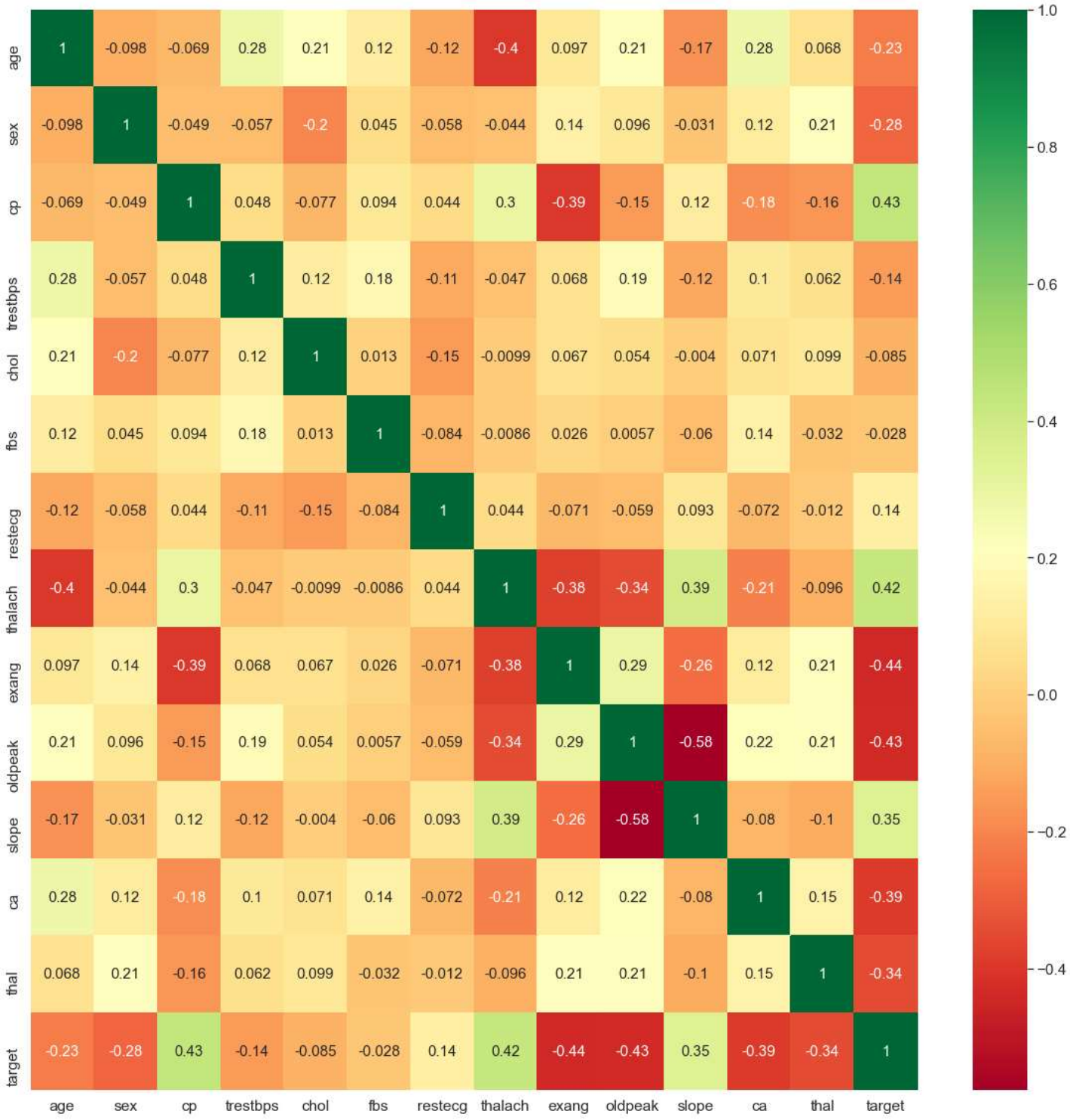
and Q1, Q2, Q3, Q4 as well as the maximum and minimum values of each column along with the number of entries which is **303** and the data type of each one of them

```
          continuous  dirty_float  low_card_int  categorical   date  \
age            False        False          True        False  False
sex            False        False         False         True  False
cp             False        False         False         True  False
trestbps        True        False         False        False  False
chol            True        False         False        False  False
fbs            False        False         False         True  False
restecg        False        False         False         True  False
thalach         True        False         False        False  False
exang          False        False         False         True  False
oldpeak         True        False         False        False  False
slope          False        False         False         True  False
ca             False        False         False         True  False
thal           False        False         False         True  False
target         False        False         False         True  False

          free_string  useless
age             False    False
sex             False    False
cp              False    False
trestbps        False    False
chol            False    False
fbs             False    False
restecg         False    False
thalach         False    False
exang           False    False
oldpeak         False    False
slope           False    False
ca              False    False
thal            False    False
target          False    False
```

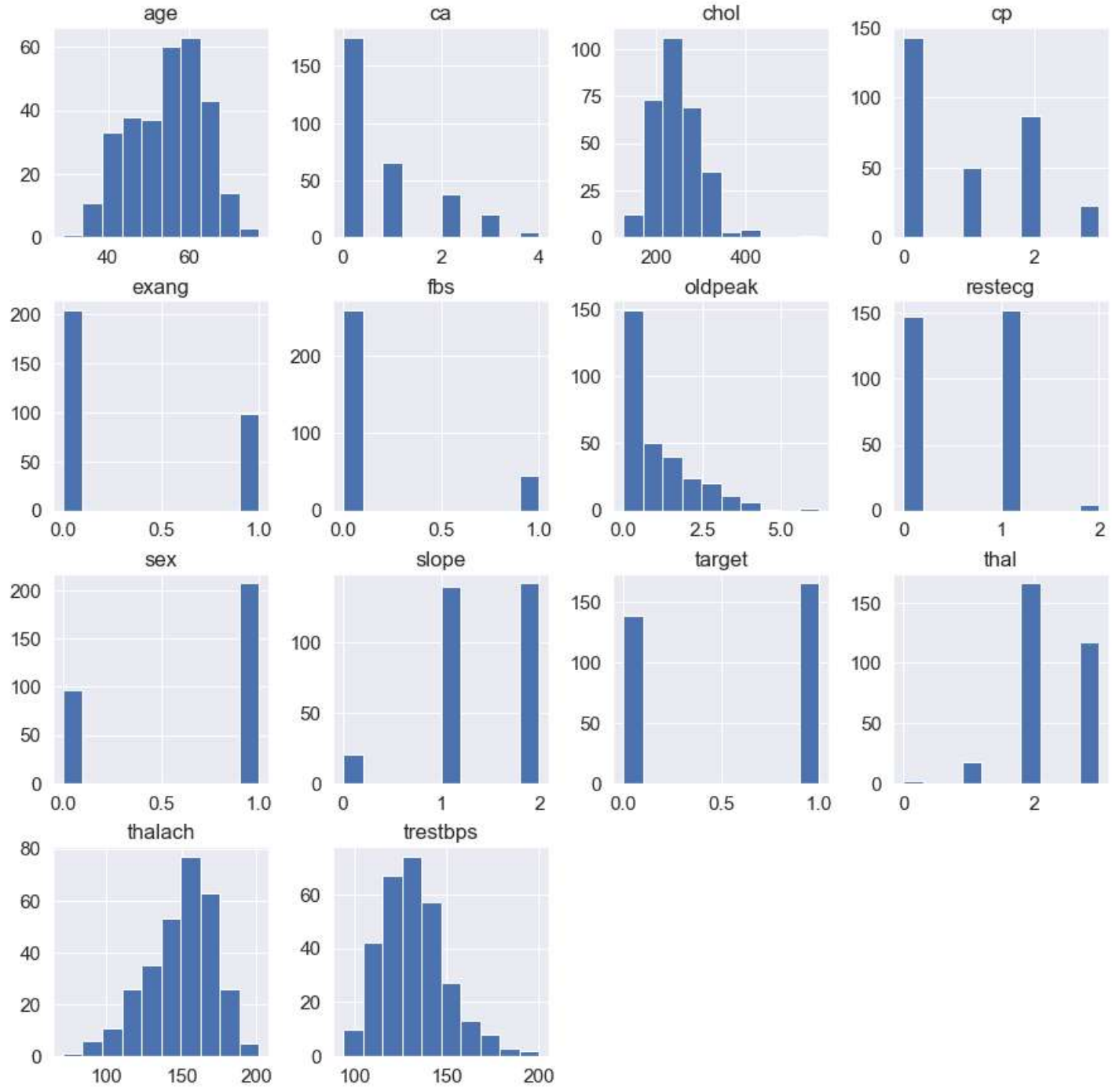
here we have a look at the varaible types given to us to understand how much cleaning of the data we will have to do before we can make our model and lucky for us there are no dirty floats freestring or useless data rather mostly categorical data which should be an easy convert into dummy variables later

Feature Selection

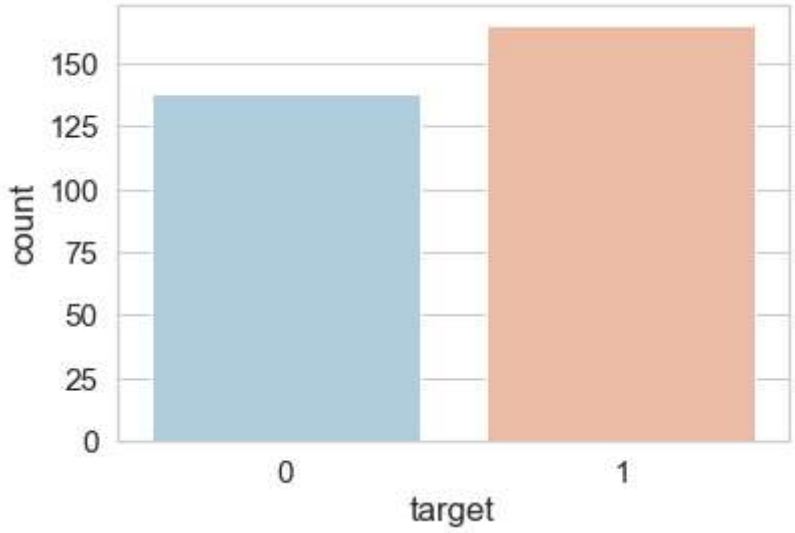


infrences from the heat map

As we can clearly see that the target variable shows **strong correlation with chest pain and maximum heart rate achieved(Thalach)** also we can see **strong negative correlation in data in exercise induced anigma (exang) and ST depression induced by exercise relative to rest (oldpeak)** so as negatively correlated factors **decline** and postively correlated factors **increase** the chances to get heart disease go up



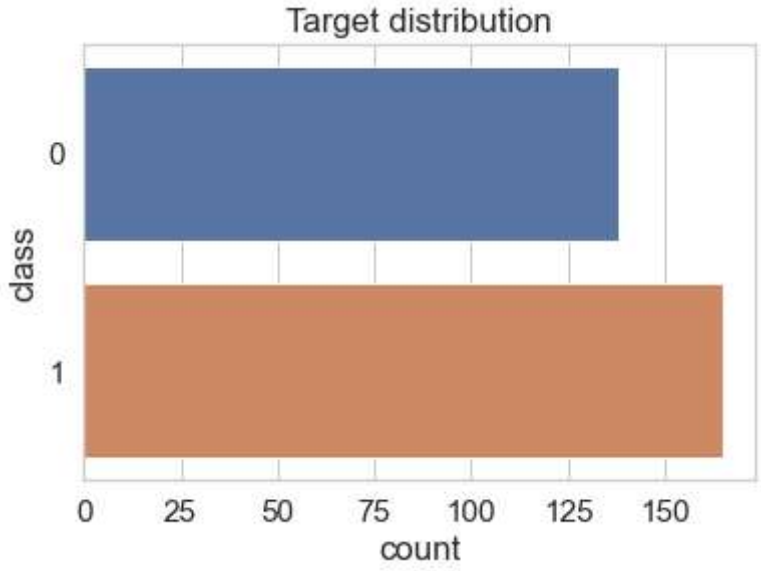
Checking if the data is balanced or not



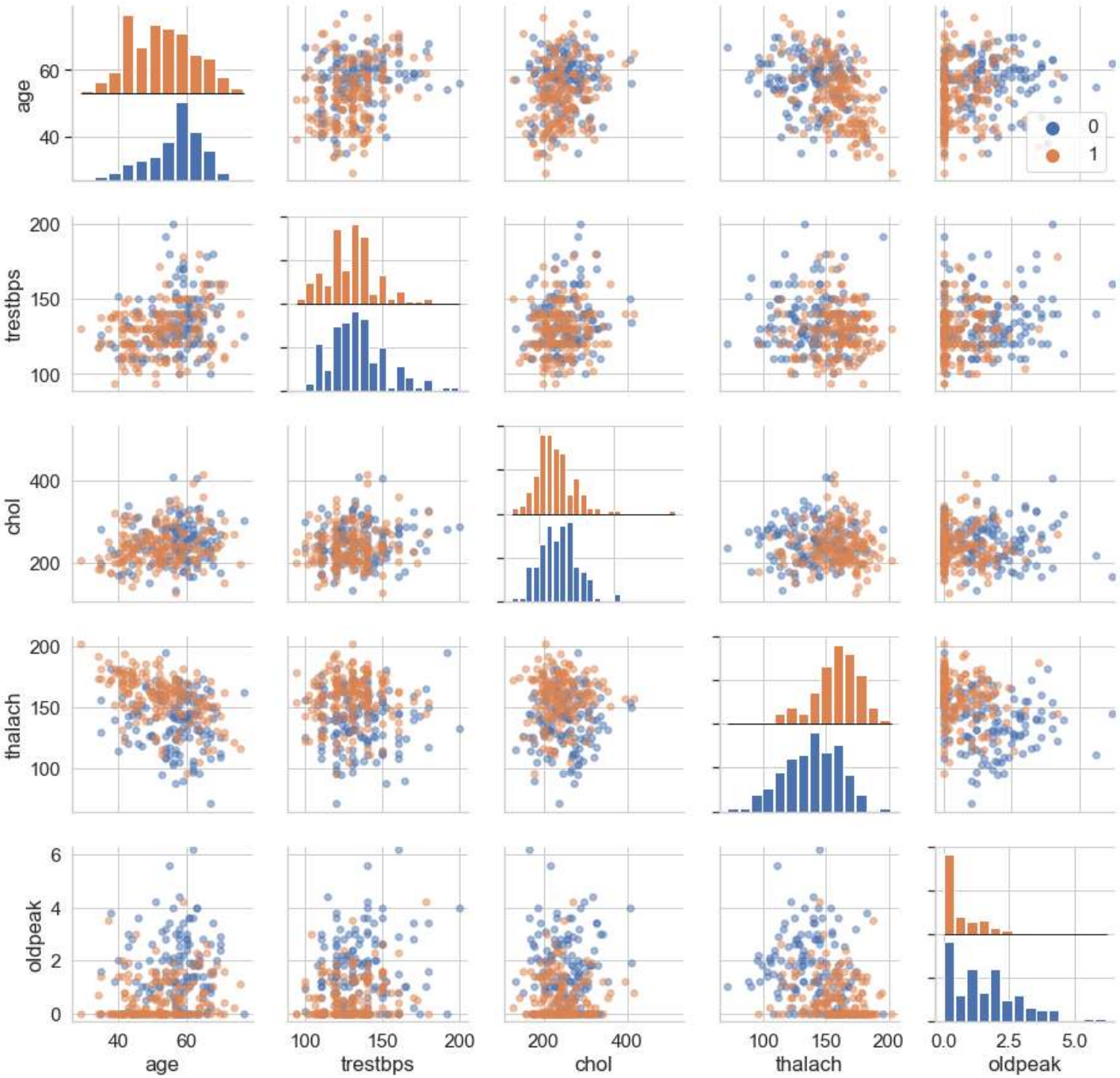
The data is almost balanced (A very good indicator)

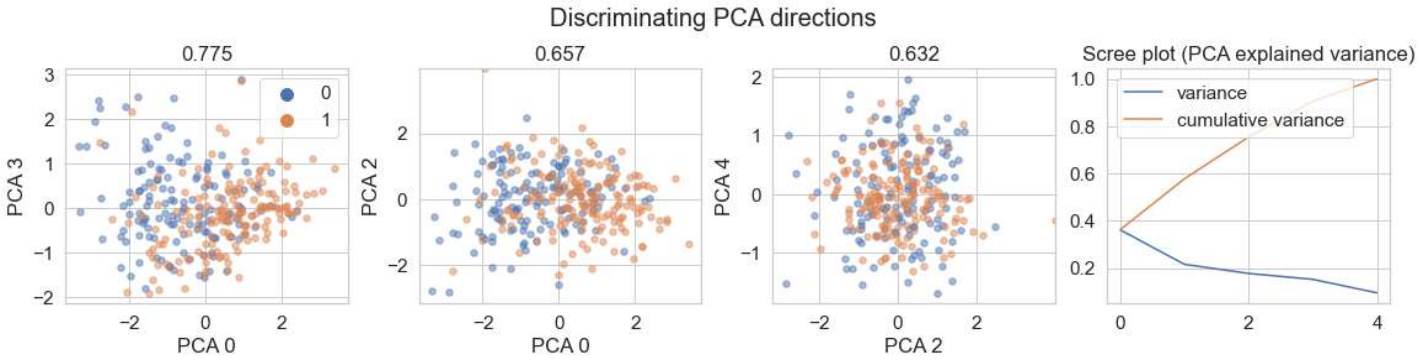
Finding the discriminating variables

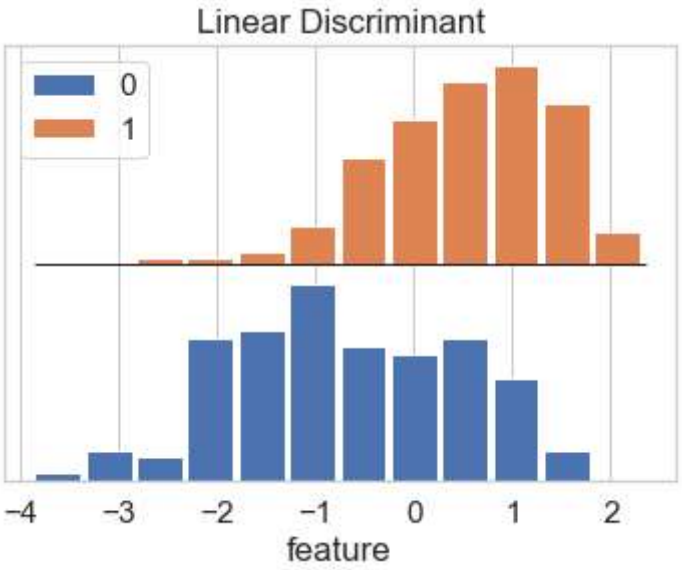
```
Target looks like classification
Linear Discriminant Analysis training set score: 0.729
```

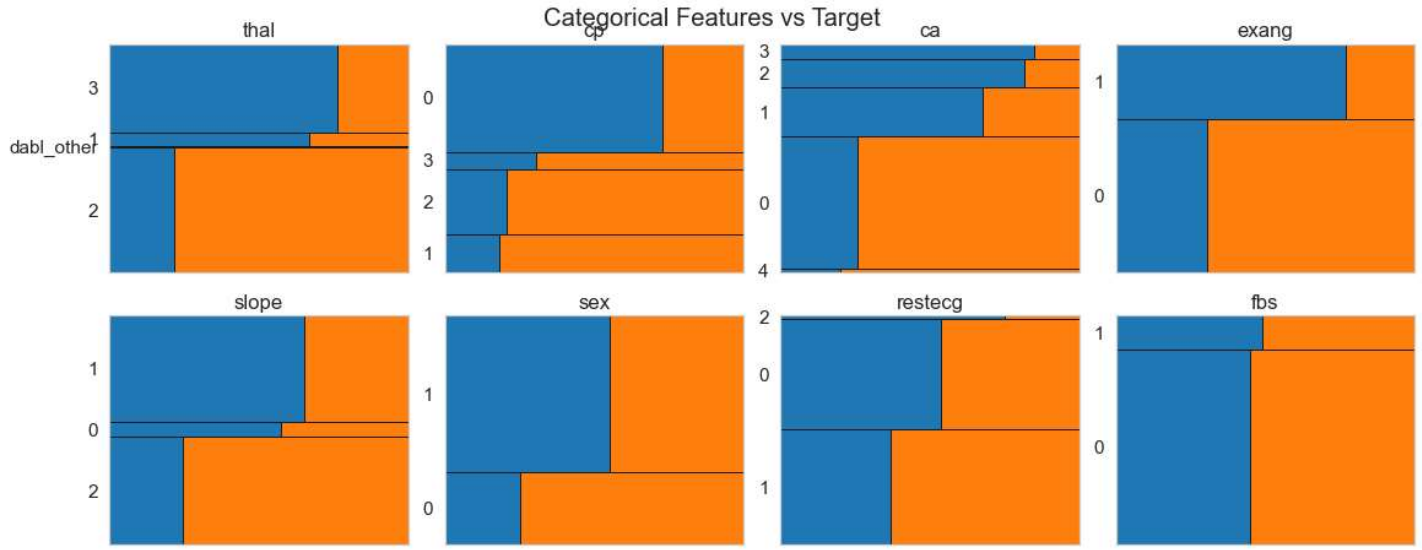


Continuous features pairplot









DATA PREPROCESSING

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1

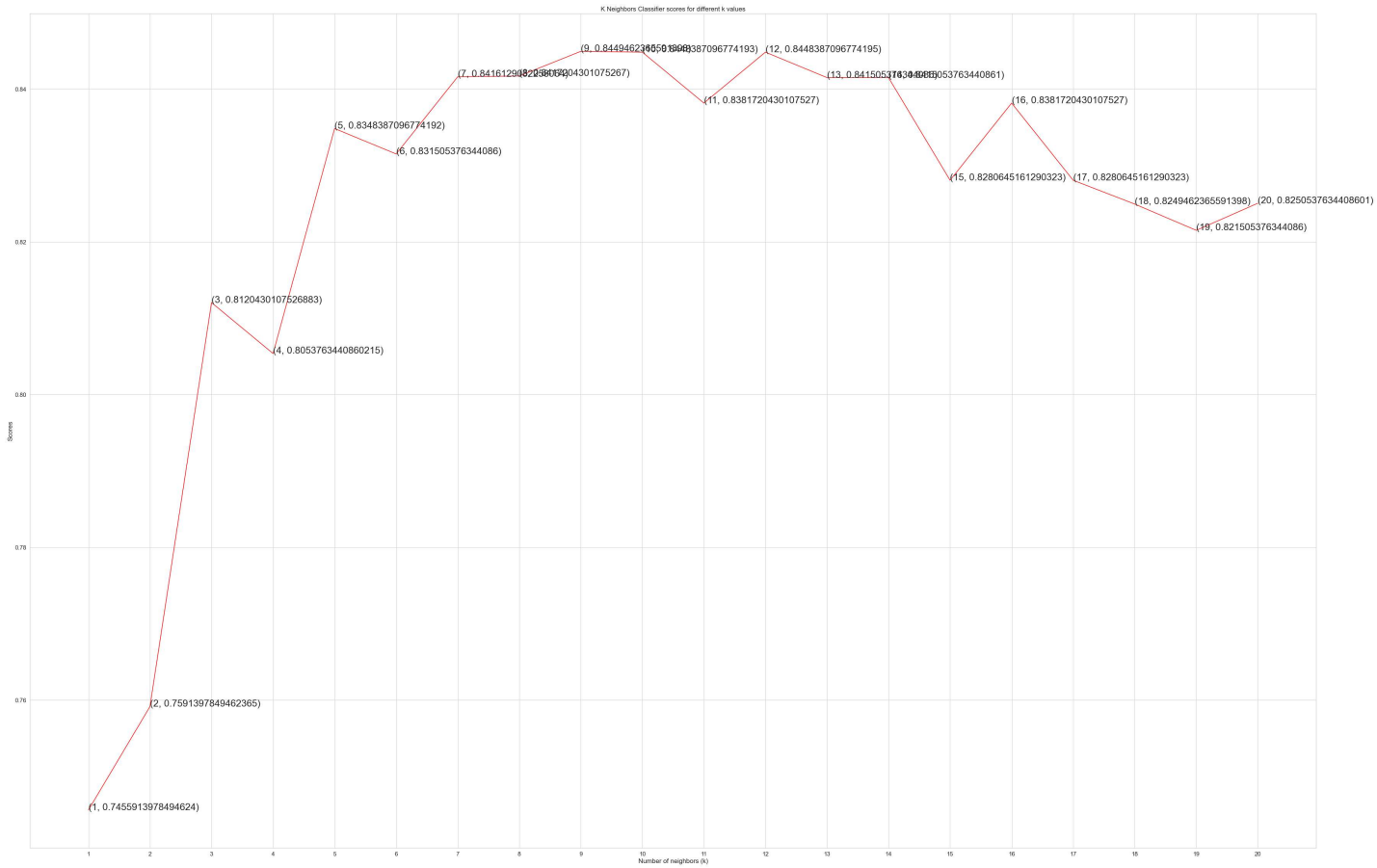
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1

creating dummy variables and std scaling

we looked at the the dataset and we see a lot of categorical variables that must be converted into dummy varaible to make more understandable for our ml model and also we need to scale these variables because many of the values under some column have been calculated with different set of precision making it hard to give accurate outputs of the model so we will use std scalar to make all these variable come down to a similar scale

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	exang_0	exang_1	slo
0	0.952197	0.763956	-0.256334	0.015443	1.087338	1	0	1	0	0	...	1	0	
1	-1.915313	-0.092738	0.072199	1.633471	2.122573	1	0	1	0	0	...	1	0	
2	-1.474158	-0.092738	-0.816773	0.977514	0.310912	1	1	0	0	1	...	1	0	
3	0.180175	-0.663867	-0.198357	1.239897	-0.206705	1	0	1	0	1	...	1	0	
4	0.290464	-0.663867	2.082050	0.583939	-0.379244	1	1	0	1	0	...	0	1	

5 rows × 31 columns




```
0.8380867498514557
```

Random Forest Classifier

```
0.8054838709677419
```

Decision Tree Classifier and Logistic Regression

I personally feel that a logistic regression would provide better accuracy for the data rather than the decision tree so we perform both using dabl

```
Running DummyClassifier(strategy='prior')
accuracy: 0.545 average_precision: 0.455 roc_auc: 0.500 recall_macro: 0.500 f1_macro: 0.353
=== new best DummyClassifier(strategy='prior') (using recall_macro):
accuracy: 0.545 average_precision: 0.455 roc_auc: 0.500 recall_macro: 0.500 f1_macro: 0.353

Running GaussianNB()
accuracy: 0.492 average_precision: 0.396 roc_auc: 0.680 recall_macro: 0.523 f1_macro: 0.437
=== new best GaussianNB() (using recall_macro):
accuracy: 0.492 average_precision: 0.396 roc_auc: 0.680 recall_macro: 0.523 f1_macro: 0.437

Running MultinomialNB()
accuracy: 0.825 average_precision: 0.299 roc_auc: 0.915 recall_macro: 0.824 f1_macro: 0.824
=== new best MultinomialNB() (using recall_macro):
accuracy: 0.825 average_precision: 0.299 roc_auc: 0.915 recall_macro: 0.824 f1_macro: 0.824

Running DecisionTreeClassifier(class_weight='balanced', max_depth=1)
```

```
accuracy: 0.693 average_precision: 0.400 roc_auc: 0.693 recall_macro: 0.693 f1_macro: 0.692
Running DecisionTreeClassifier(class_weight='balanced', max_depth=5)
accuracy: 0.759 average_precision: 0.370 roc_auc: 0.775 recall_macro: 0.757 f1_macro: 0.757
Running DecisionTreeClassifier(class_weight='balanced', min_impurity_decrease=0.01)
accuracy: 0.772 average_precision: 0.365 roc_auc: 0.789 recall_macro: 0.771 f1_macro: 0.770
Running LogisticRegression(C=0.1, class_weight='balanced', max_iter=1000)
accuracy: 0.838 average_precision: 0.296 roc_auc: 0.916 recall_macro: 0.837 f1_macro: 0.837
=== new best LogisticRegression(C=0.1, class_weight='balanced', max_iter=1000) (using recall_macr
o):
accuracy: 0.838 average_precision: 0.296 roc_auc: 0.916 recall_macro: 0.837 f1_macro: 0.837

Running LogisticRegression(class_weight='balanced', max_iter=1000)
accuracy: 0.852 average_precision: 0.296 roc_auc: 0.913 recall_macro: 0.851 f1_macro: 0.850
=== new best LogisticRegression(class_weight='balanced', max_iter=1000) (using recall_macro):
accuracy: 0.852 average_precision: 0.296 roc_auc: 0.913 recall_macro: 0.851 f1_macro: 0.850


Best model:
LogisticRegression(class_weight='balanced', max_iter=1000)
Best Scores:
accuracy: 0.852 average_precision: 0.296 roc_auc: 0.913 recall_macro: 0.851 f1_macro: 0.850
```

END HYPOTHESIS

After analysing all the classifier and their accuracy we came to the conclusion that logistic regression was the best predictor for the heart stroke and with highst accuracy of 85.2% percent of the target data hence predicting if the patient is suffering from heart strong based on variable provided.

SUMMARY

In this project I did the following tasks

- imported the data repository from kaggle
- imported the specific libraries
- Analysed the dataset calculated it's mean,min, max, count, Q1,Q2,Q3,Q4 and std using "describe()"
- checked for if any variables required cleaning by finding their datat types using "dabl"
- Performed feature selection where i made a heatmap of the dataset to see the relation between target and other variables using "seaborn"
- Made infrences from the heatmap
- Made a histogram and chekced if the data was balanced or not using "Seaborn"
- Found target distribution, found the Discriminating PCA direction and the relation of all categorical variable to the target variable using "dabl"
- Created dummy variables for the models and performed standard scaling using "sklearn and "StandardScaler"
- Seperated the data into training and testing using "train_test_split()" from "sklearn"
- Made a working Knn classifier on the data set and used cross validation for improved accuracy using "sklearn" and "KNeighborsClassifier()"
- Made a random forest classifier and compared it's accuracy with the knn classifier using "sklearn" and "RandomForestClassifier()"

- Made a desicion tree classifier and a performed logitic regression then checked the accuracy of each and present the results of the model with the highest accuracy which was logistic regression using "dabl"