Heart Disease

Done by : Rawan Al-Qahtani

Dr : Mejdal Al-Qahtani

Introduction

This project is to predict whether the patient has a **10-year risk of future coronary heart disease (CHD)**. The data was taken from Kaggle website as (.csv)file, and it is from a cardiovascular study on residents of the town of Framingham, Massachusetts.

After features checked and data splitting a comparison between (KNN – logistic regression - Random forest - Decision tree) Classifiers was made. First, the hypermeters where found then (Accuracy- Recall-Precision -F1- Confusion Matrix) for classifiers was shown. Then, the most compatible classifier was chosen to be the model for prediction which is **Logistic regression** since it had the highest recall score of 66.9% with the up-sampled data.

Extract The 10 Best Features

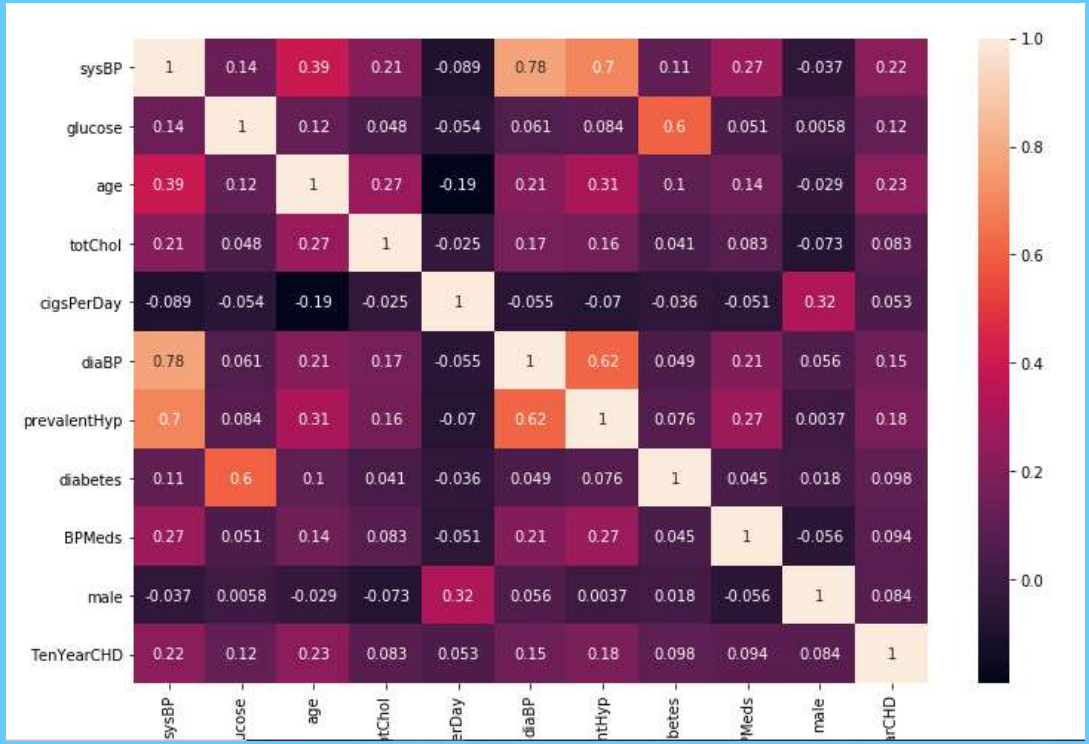
The highest features with effected people

81]:
```
#extract the 10 best features
features = SelectKBest(score_func=chi2, k=10)
scores = features.fit(x,y).scores_
columns = x.columns

df = pd.DataFrame()
df['score'] = scores
df['column'] = columns

best_feat = df.sort_values(by='score', ascending=False).head(10)
best_feat
```

81]:

	score	column
10	728.028608	sysBP
14	370.026234	glucose
1	318.439180	age
9	233.735513	totChol
4	178.115009	cigsPerDay
11	151.897052	diaBP
7	92.181857	prevalentHyp
8	38.371005	diabetes
5	35.323423	BPMeds
0	16.707199	male

Correlation Heatmap



Data splitting & Sampling

Data splitting

[82]:
```
split = StratifiedShuffleSplit(n_splits = 1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(heart_scaled, heart_scaled['TenYearCHD']):
    train = heart_scaled.loc[train_index]
    test = heart_scaled.loc[test_index]

    x_test = test.drop("TenYearCHD", axis=1)
    y_test = test["TenYearCHD"]
```

Sampling

[83]:
```
#Separate majority and minority classes
majority = train[train["TenYearCHD"] == 0]
minority = train[train["TenYearCHD"] == 1]
```

[84]:
```
#upsample minority data
minority_upsampled = resample(minority,
                              n_samples =2500,
                              random_state=65)

#combine with mojority dataframe
upsampled = pd.concat([majority, minority_upsampled],axis=0)
```

]:
```
#downsample majority data
majority_downsampled = resample(majority,
                                n_samples=511,
                                random_state=65)

downsampled = pd.concat([minority,majority_downsampled],axis=0)
```

```
Upsampled data:
 0.0    2804
1.0    2500
Name: TenYearCHD, dtype: int64
-----
Downsampled data:
 0.0    511
1.0    502
Name: TenYearCHD, dtype: int64
```

Choosing The Best Hyperparameters

Classifier	The best score / Upsampled	The best score / Downsampled
KNN	0.9245852187028658	0.6811451135241856
Logistic regression	0.6830693815987934	0.6712734452122409
Random forest	0.8374811463046757	0.6712734452122409
Decision tree	0.8849924585218703	0.6238894373149062

Modelling

Accuracy

Logistic Regression	0.7230955259975816
KNN	0.7678355501813785
Decision Tree	0.7412333736396615
Random Forest	0.7799274486094316

Recall

Logistic Regression	0.6031746031746031
KNN	0.18253968253968253
Decision Tree	0.23809523809523808
Random Forest	0.4126984126984127

the logistic regression has the best parameters spicily the recall

Precision

Logistic Regression	0.2980392156862745
KNN	0.20535714285714285
Decision Tree	0.20270270270270271
Random Forest	0.325

F1

Logistic Regression	0.3989501312335958
KNN	0.19327731092436976
Decision Tree	0.21897810218978103
Random Forest	0.3636363636363636

CONFUSION MATRIX

CONFUSION MATRIX

In [99]:

```
print('Logistic Regression:\n ', confusion_matrix(y_test, lr.predict(x_test)))
print('---------')
print('KNN: \n', confusion_matrix(y_test, knn.predict(x_test)))
print('---------')
print('Decision Tree: \n', confusion_matrix(y_test, tree.predict(x_test)))
print('---------')
print('Random Forest: \n', confusion_matrix(y_test, clf.predict(x_test)))
print('---------')
```

```
Logistic Regression:
  [[522 179]
 [ 50  76]]
---------
KNN:
 [[612  89]
 [103  23]]
---------
Decision Tree:
 [[583 118]
 [ 96  30]]
---------
Random Forest:
 [[593 108]
 [ 74  52]]
---------
```

Also here at the matrix the logistic tree has the least FP and higher TN

Module trining for data prdication

```python
#intialize model
lr = LogisticRegression()
```

```python
def predict_80_20(df, target):
    split = StratifiedShuffleSplit(n_splits = 1, test_size=0.2, random_state=42)
    for train_index, test_index in split.split(df, df[target]):
        train = df.loc[train_index]
        test = df.loc[test_index]

        x = train.drop([target], axis=1)
        y = train[target]

        a = test.drop([target], axis=1)
        b = test[target]

        lr.fit(x,y)

        predicted = lr.predict(a)
        accuracy = accuracy_score(b, predicted)
        c_m = confusion_matrix(b, predicted)

        return print('Accuracy score: ', accuracy, '\n', 'Confusion matrix: \n', c_m)
```

Prepared as function to be called later

Key Numbers

```
2]: 1 %run predict_CHD.ipynb # calling the  LogisticRegression
```

```
3]: 1 predict_80_20(heart1, 'TenYearCHD')
```

```
Accuracy score:  0.8524788391777509
 Confusion matrix:
 [[697   4]
 [118   8]]
```

Drop education column

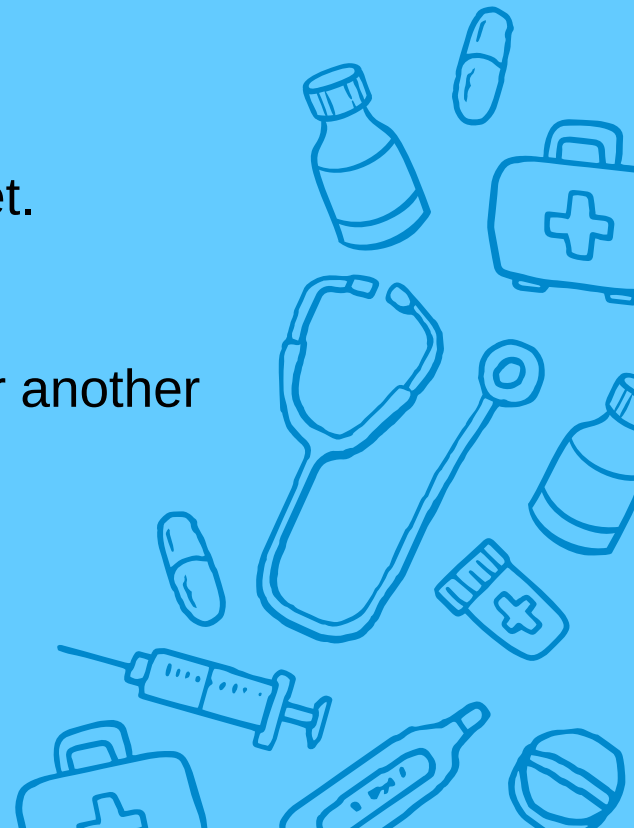
```
1 heart_ed = heart1.drop(['education'], axis = 1)
```

```
1 predict_80_20(heart_ed, 'TenYearCHD')
```

```
Accuracy score:  0.8512696493349455
 Confusion matrix:
 [[697   4]
 [119   7]]
```

Future work

- Enhancing the model with larger dataset.
- Using some columns to make model for another disease



Thanks