AssessmentProject2

October 18, 2022

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import␣
 ↪mean_squared_error,accuracy_score,confusion_matrix,classification_report
from math import sqrt

import warnings
warnings.filterwarnings("ignore")
```

0.0.1 1.Preliminary analysis

[ ]:
```
cep1_data = pd.read_excel('1645792390_cep1_dataset.xlsx')
cep1_data.head()
```

[ ]:
```
cep1_data.shape
```

[ ]:
```
cep1_data.isna().sum()
```

[ ]:
```
cep1_data.drop_duplicates(inplace=True)
cep1_data.duplicated().sum()
```

0.0.2 2. Distribution of the disease and the related factors

[ ]:
```
cep1_data.describe()
```

[ ]:
```
cep1_data.select_dtypes(include='object')
```

[ ]:
```
sns.set(rc = {'figure.figsize':(5,5)})
for col in cep1_data:
    print (col,'\n',cep1_data[col].value_counts())
```

```
    sns.countplot(data=cep1_data, x=col)
    plt.show()
```

[ ]:
```
cep1_data.groupby('sex').mean()
```

[ ]:
```
cep1_data.groupby('age')[['target']].count()
```

[ ]:
```
cep1_data.plot(kind='scatter',x='trestbps',y='target')
plt.show()
cep1_data.target.corr(cep1_data.trestbps)
```

There is weak correlation between target and resting blood pressure (trestbps) of a patient

[ ]:
```
# relationship between cholesterol levels and a target variable
cep1_data.plot(kind='scatter',x='chol',y='target')
plt.show()
print ('correlation between target and cholesterol levels :', cep1_data.target.
 ↪corr(cep1_data.chol))
```

There is a weak correlation between target and cholesterol levels

relationship exists between peak exercising and the occurrence of a heart attack

[ ]:
```
# relationship between thalach(maximum heart rate achived) and slope (slope of␣
 ↪peak exercise)
cep1_data.plot(kind='scatter',x='slope',y='thalach')
plt.show()
print ('correlation between maximum heart rate and peak exercise :', cep1_data.
 ↪thalach.corr(cep1_data.slope))
```

[ ]:
```
#relationship between thalassemia(thal) and target
cep1_data.plot(kind='scatter',x='thal',y='target')
plt.show()
print ('correlation between thalassemia(thal) and target :', cep1_data.target.
 ↪corr(cep1_data.thal))
```

[ ]:
```
#factors determine the occurrence of CVD
#drawing heatmap of correlations
sns.set(rc = {'figure.figsize':(15,15)})
sns.heatmap(cep1_data.corr(),annot=True,cmap='Blues')
plt.show()
sns.set(rc = {'figure.figsize':(5,5)})
```

There is a correlation between Target v CP , Target v thalach and Target v slope

[ ]:
```
#pairplot to understand the relationship between all the given variables

sns.pairplot(cep1_data[cep1_data.target.isin([1])],hue='target')
plt.show()
```

0.0.3 3. baseline model to predict the risk of a heart attack

3. Build a baseline model to predict the risk of a heart attack using a logistic regression and random forest and explore the results while using correlation analysis and logistic regression (leveraging standard error and p-values from statsmodels) for feature selection

Models with all input variables

[ ]:
```
#preparing train data  and test data

y = cep1_data['target']
x = cep1_data.drop(columns='target',axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪3,random_state=42)
```

[ ]:
```
#standarizing training and dataset
x_train = (x_train-x_train.mean())/x_train.std()
x_test = (x_test-x_test.mean())/x_test.std()
y_train = (y_train-y_train.mean())/y_train.std()
y_test = (y_test-y_test.mean())/y_test.std()
```

[ ]:
```
# Random Forest Regression
regressor =RandomForestRegressor(n_estimators=100)
regressor.fit( x_train,y_train)
y_prediction=regressor.predict(x_test)
print ('RandomForest Model Accuracy =',regressor.score(x_test,y_test))
```

[ ]:
```
#since logisticRegression() do not take continuous data, therefore, performing␣
 ↪label encoding in order to normalise the target variable
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.fit_transform(y_test)
```

[ ]:
```
#logistic regression
regressor =LogisticRegression()
regressor.fit( x_train,y_train)
y_prediction=regressor.predict(x_test)

#Accuracy
print ('Logistic Model Accuracy =',regressor.score(x_test,y_test))
```

```
#RMSE
print ('Logistic Model RMSE =',mean_squared_error(y_test, y_prediction))

#confusion matrix
logistic_cm=confusion_matrix(y_test,y_prediction)
logistic_cm
```

Models with highly dependent input variables

[ ]:
```
#finding input variables having high correlation with target

sns.set(rc = {'figure.figsize':(15,15)})
sns.heatmap(cep1_data.corr(),annot=True,cmap='Blues')
plt.show()
sns.set(rc = {'figure.figsize':(5,5)})
```

[ ]:
```
cep1_data.columns
```

[ ]:
```
corr =abs(cep1_data.corr())['target']
corr[corr>0.25]
```

[ ]:
```
#preparing train data  and test data
highCorrVariable=['sex','cp','thalach','exang','oldpeak','ca','slope','thal']
y = cep1_data['target']
x = cep1_data[highCorrVariable]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪3,random_state=42)
```

[ ]:
```
#standarizing training and dataset
x_train = (x_train-x_train.mean())/x_train.std()
x_test = (x_test-x_test.mean())/x_test.std()
y_train = (y_train-y_train.mean())/y_train.std()
y_test = (y_test-y_test.mean())/y_test.std()
```

[ ]:
```
# Random Forest Regression
regressor =RandomForestRegressor(n_estimators=100)
regressor.fit( x_train,y_train)
y_prediction=regressor.predict(x_test)
print ('RandomForest Model Accuracy =',regressor.score(x_test,y_test))
```

[ ]:
```
#since logisticRegression() do not take continuous data, therefore, performing␣
↪label encoding in order to normalise the target variable
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.fit_transform(y_test)
```

[ ]:
```
#logistic regression
regressor =LogisticRegression()
regressor.fit( x_train,y_train)
y_prediction=regressor.predict(x_test)

#Accuracy
print ('Logistic Model Accuracy =',regressor.score(x_test,y_test))

#RMSE
print ('Logistic Model RMSE =',mean_squared_error(y_test, y_prediction))

#confusion matrix
logistic_cm=confusion_matrix(y_test,y_prediction)
logistic_cm
```

[ ]:

[ ]: