Capstone 2project Health Care (1)

May 16, 2023

[ ]:

[ ]:
```
#import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
import seaborn as sns
warnings.filterwarnings(action = "ignore", category = FutureWarning)
```

[ ]:
```
data= pd.read_csv("health care diabetes.csv")
data
```

0.0.1 1. Perform descriptive analysis. It is very important to understand the variables and corresponding values. We need to think through - Can minimum value of below listed columns be zero (0)? On these columns, a value of zero does not make sense and thus indicates missing value.Glucose, BloodPressure, SkinThickness, Insuline, BMI.How will you treat these values?

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

[ ]:
```
data.head(15)
```

0.0.2 we can see there are 0 in the columns of bloodpressure, bmi, insulin etc.. so this value doesnot make any sense. so replacing all the 0 value with median.

[ ]:
```
data['Glucose']=data['Glucose'].replace(0,data['Glucose'].median())
data['BloodPressure']=data['BloodPressure'].replace(0,data['BloodPressure'].
 ↪median())
data['SkinThickness']=data['SkinThickness'].replace(0,data['SkinThickness'].
 ↪median())
data['Insulin']=data['Insulin'].replace(0,data['Insulin'].median())
data['BMI']=data['BMI'].replace(0,data['BMI'].median())
```

```
data.describe()
```

0.0.3 2 Visually explore these variables using histograms. Treat the missing values accordingly.

[ ]:
```
data.isna().any()
```

[ ]:
```
plt.hist("BloodPressure", data= data)
```

[ ]:
```
plt.hist("Glucose", data= data)
```

[ ]:
```
plt.hist("SkinThickness", data= data)
```

[ ]:
```
plt.hist("Insulin", data= data)
```

[ ]:
```
plt.hist("BMI", data= data)
```

[ ]:
```
fig, ax= plt.subplots(ncols= 3, figsize=(15,5))

sns.histplot(x= "Pregnancies", data= data, ax=ax[0])
sns.histplot(x= "Glucose", data= data, ax= ax[1])
sns.histplot(x= "BloodPressure", data= data, ax= ax[2])
```

[ ]:
```
fig, ax= plt.subplots(ncols=2 , figsize=(15,5))
sns.histplot(x= "Insulin", data= data, ax= ax[0])
sns.histplot(x= "BMI", data= data, ax= ax[1])
```

0.0.4 3. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

[ ]:
```
data.dtypes
```

[ ]:
```
fig, ax= plt.subplots(ncols=3, figsize= (10,5))
sns.countplot(x= "Pregnancies", data= data, ax= ax[0])
sns.countplot(x= "Glucose", data= data, ax= ax[1])
sns.countplot(x= "BloodPressure", data= data, ax= ax[2])
plt.tight_layout()
```

0.0.5 4. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.

[ ]:
```
sns.countplot(x= "Outcome", data= data)
```

[ ]:
```
print("value of \n", data["Outcome"].value_counts())
```

[ ]:
```
data["Outcome"].value_counts()/len(data)
```

0.0.6 The data set is balanced.

0.0.7 5. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.

[ ]:
```
sns.pairplot(data)
```

0.0.8 with this visualisation. we can see positive correlation between BMI and Skin Thickness ; and also between age and pregnancy,

[ ]:
```
data.corr()
```

[ ]:
```
plt.subplots(figsize=(15,9))
sns.heatmap(data.corr(), annot= True)
```

1 Project Task: Week 2

1.1 Data Modeling:

1.1.1 1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.

Apply an appropriate classification algorithm to build a model.

Compare various models with the results from KNN algorithm.

Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc.

Please be descriptive to explain what values of these parameter you have used.

1.1.2 The logistic regression will suit best as our dependent value is categorical data and independent variable are continuos.

[ ]:
```
#training and test data
X= data[["Pregnancies","Glucose", 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']]
y= data[["Outcome"]]

#Importing "train_test-split" function to test the model
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

[ ]:
```
print("shape of train data is", X_train.shape)
print("shape of test data is", X_test.shape)
```

[ ]:
```
#importing Logistic Regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

#Fit the model in train and test data
lr.fit(X_train,y_train).score(X_train,y_train)
```

[ ]:
```
#Now fitting the model in test set
prediction=lr.predict(X_test)
```

[ ]:
```
#Printing first 5 rows after fitting the model in test set
print (X_test.head())
```

[ ]:
```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, prediction)
print('\n', "Confusion metrics is ",'\n', cm, '\n')
accuracy = metrics.accuracy_score(y_test, prediction)
print( '\n', "Accuracy score of logistic regression is :",accuracy, '\n')

print ( "classification score is",'\n',   metrics.classification_report(y_test,␣
 ↪prediction))
```

1.1.3 Logistic Regression gives 75% acccuracy

[ ]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
lr_roc_auc = roc_auc_score(y_test, lr.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, lr.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % lr_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
print('AUC: %.3f' % lr_roc_auc)
plt.show()
```

2 SVM Model

[ ]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

[ ]:
```
model=SVC()
model.fit(X_train, y_train)
#creating the predictions on the test data i.e. X_test
y_pred =model.predict(X_test)
accuracy= accuracy_score(y_test, y_pred)
print("Accuracy for the SVM Classifier: {:.3f}".format(accuracy)) #.3f is upto␣
↪3 places decimal
```

[ ]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
lr_roc_auc = roc_auc_score(y_test, lr.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, lr.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % lr_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
```

```
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
print('AUC: %.3f' % lr_roc_auc)
plt.show()
```

3 KNN Model

[ ]:
```
#feature Scaling
from sklearn.preprocessing import StandardScaler
st_X= StandardScaler()
X_train= st_X.fit_transform(X_train)
X_test= st_X.transform(X_test)
```

[ ]:
```
#Fitting K-NN classifier to the training set
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(X_train, y_train)
```

[ ]:
```
#Predicting the test set result
y_pred= classifier.predict(X_test)
```

[ ]:
```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix , classification_report
cm= confusion_matrix(y_test, y_pred)
cm
```

[ ]:
```
print(classification_report(y_test, y_pred))
```

[ ]:
```
accuracy= accuracy_score(y_test, y_pred)
print("Accuracy for KNN model is: {:.2f}".format(accuracy)) #.2f is upto 2␣
 ↪places decimal
```

4 Random Forest

[ ]:
```
#fitting model
from sklearn.ensemble import RandomForestClassifier
clf= RandomForestClassifier(criterion="gini", #gini is entropy , y we use? we␣
 ↪see how much gini index reducing for this
                            max_depth= 7,
                            n_estimators= 200,
                            random_state= 5)
```

```
#the min. no. of trees in the forest should be atleast 40-50 , so n_estimator␣
 ↪=200 or nything more then 50 u can take
# the gini index heps us to understand the highest IG or the lowest entropy of␣
 ↪each feature. So essentially
# it will help us select the most imp variablles in our model.
# max depth allow is 3, 5,7)
```

[ ]:
```
#fitting the training data
clf.fit(X_train, y_train)
```

[ ]:
```
clf.feature_importances_
```

[ ]:
```
data.columns
```

5 Glucose is very important as the value is max

[ ]:
```
y_pred= clf.predict(X_test)
```

[ ]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

[ ]:
```
#find accuracy score

from sklearn.metrics import accuracy_score
print("Accuracy of Random Forest is ", accuracy_score(y_test, y_pred))
```

6 decision Tree

[ ]:
```
#Fitting a decision tree clasifier
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

[ ]:
```
#test the accuracy of the decision tree
predictions=dtree.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,predictions))
```

[ ]:
```
from sklearn.metrics import accuracy_score
print("Accuracy of Decision Tree is ", accuracy_score(y_test, predictions))
```