**#ml4 diabetes**

**import pandas as pd**

**import numpy as np**

**data = pd.read\_csv("diabetes.csv")**

**data.head()**

**data.isnull().any()**

**data.describe().T**

**data\_copy = data.copy(deep = True)**

**data\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = data\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)**

**data\_copy.isnull().sum()**

**p = data.hist(figsize = (20,20))**

**data\_copy['Glucose'] = data\_copy['Glucose'].fillna(data\_copy['Glucose'].mean())**

**data\_copy['BloodPressure'] = data\_copy['BloodPressure'].fillna(data\_copy['BloodPressure'].mean())**

**data\_copy['SkinThickness'] = data\_copy['SkinThickness'].fillna(data\_copy['SkinThickness'].median())**

**data\_copy['Insulin'] = data\_copy['Insulin'].fillna(data\_copy['Insulin'].median())**

**data\_copy['BMI'] = data\_copy['BMI'].fillna(data\_copy['BMI'].median())**

**p = data\_copy.hist(figsize = (20,20))**

**import missingno as msno**

**p = msno.bar(data)**

**p=data.Outcome.value\_counts().plot(kind="bar")**

**import seaborn as sns**

**p=sns.pairplot(data\_copy, hue = 'Outcome')**

**import matplotlib.pyplot as plt**

**plt.figure(figsize=(12,10))**

**p=sns.heatmap(data.corr(), annot=True,cmap ='RdYlGn')**

**plt.figure(figsize=(12,10)) # on this line I just set the size of figure to 12 by 10.**

**p=sns.heatmap(data\_copy.corr(), annot=True,cmap ='RdYlGn')**

**from sklearn.preprocessing import StandardScaler**

**sc\_X = StandardScaler()**

**X = pd.DataFrame(sc\_X.fit\_transform(data\_copy.drop(["Outcome"], axis =1),),columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',**

**'X.head()BMI', 'DiabetesPedigreeFunction', 'Age'])**

**y =data\_copy.Outcome**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 42, stratify=y)**

**from sklearn.neighbors import KNeighborsClassifier**

**train\_scores = []**

**test\_scores = []**

**for i in range(1,15):**

**knn = KNeighborsClassifier(i)**

**knn.fit(X\_train, y\_train)**

**train\_scores.append(knn.score(X\_train, y\_train))**

**test\_scores.append(knn.score(X\_test, y\_test))**

**max\_test\_score =max(test\_scores)**

**test\_score\_index = [i for i, v in enumerate(test\_scores) if v== max\_test\_score]**

**print('Max test score {} % and k = {}'.format(max\_test\_score\*100,list(map(lambda x: x+1, test\_score\_index))))**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# Example data for train\_scores and test\_scores**

**train\_scores = [0.8, 0.82, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96]**

**test\_scores = [0.75, 0.76, 0.78, 0.79, 0.8, 0.82, 0.83, 0.84, 0.85, 0.84, 0.83, 0.82, 0.81, 0.8]**

**# Create a figure with specified size**

**plt.figure(figsize=(12, 5))**

**# Use keyword arguments for x and y**

**p = sns.lineplot(x=range(1, 15), y=train\_scores, marker='\*', label='Train Score')**

**p = sns.lineplot(x=range(1, 15), y=test\_scores, marker='o', label='Test Score')**

**# Add titles and labels**

**plt.title('Train and Test Scores')**

**plt.xlabel('Number of Samples')**

**plt.ylabel('Scores')**

**plt.legend()**

**# Show the plot**

**plt.show()**

**knn = KNeighborsClassifier(11)**

**knn.fit(X\_train,y\_train)**

**knn.score(X\_test,y\_test)**

**from mlxtend.plotting import plot\_decision\_regions**

**value = 20000**

**width =20000**

**plot\_decision\_regions(X.values, y.values, clf = knn, legend =2,filler\_feature\_values={2: value, 3: value, 4: value, 5: value, 6: value, 7: value},**

**filler\_feature\_ranges={2: width, 3: width, 4: width, 5: width, 6: width, 7: width},**

**X\_highlight=X\_test.values)**

**plt.title("KNN with diabetes data")**

**plt.show()**

**from sklearn.metrics import confusion\_matrix**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, fbeta\_score**

**y\_pred = knn.predict(X\_test)**

**cnf\_matrix = confusion\_matrix(y\_test, y\_pred)**

**p = sns.heatmap(pd.DataFrame(cnf\_matrix), annot=True, cmap="YlGnBu" ,fmt='g')**

**plt.title('Confusion matrix', y=1.1)**

**plt.ylabel('Actual label')**

**plt.xlabel('Predicted label')**

**def model\_evaluation(y\_test, y\_pred, model\_name):**

**acc = accuracy\_score(y\_test, y\_pred)**

**prec = precision\_score(y\_test, y\_pred)**

**rec = recall\_score(y\_test, y\_pred)**

**f1 = f1\_score(y\_test, y\_pred)**

**f2 = fbeta\_score(y\_test, y\_pred, beta = 2.0)**

**results = pd.DataFrame([[model\_name, acc, prec, rec, f1, f2]],**

**columns = ["Model", "Accuracy", "Precision", "Recall",**

**"F1 SCore", "F2 Score"])**

**results = results.sort\_values(["Precision", "Recall", "F2 Score"], ascending = False)**

**return results**

**model\_evaluation(y\_test, y\_pred, "KNN")# Alternate way**

**from sklearn.metrics import classification\_report**

**print(classification\_report(y\_test,y\_pred))**

**precision recall f1-score support**

**0 0.80 0.85 0.83 167**

**1 0.68 0.61 0.64 89**

**accuracy 0.77 256**

**macro avg 0.74 0.73 0.73 256**

**weighted avg 0.76 0.77 0.76 256**

**from sklearn.metrics import classification\_report**

**print(classification\_report(y\_test,y\_pred))**

**from sklearn.metrics import auc, roc\_auc\_score, roc\_curve**

**y\_pred\_proba = knn.predict\_proba(X\_test)[:,-1]**

**fpr, tpr, threshold = roc\_curve(y\_test, y\_pred\_proba)**

**classifier\_roc\_auc = roc\_auc\_score(y\_test, y\_pred\_proba)**

**plt.plot([0,1],[0,1], label = "---")**

**plt.plot(fpr, tpr, label ='KNN (area = %0.2f)' % classifier\_roc\_auc)**

**plt.xlabel("fpr")**

**plt.ylabel("tpr")**

**plt.title('Knn(n\_neighbors=11) ROC curve')**

**plt.legend(loc="lower right", fontsize = "medium")**

**plt.xticks(rotation=0, horizontalalignment="center")**

**plt.yticks(rotation=0, horizontalalignment="right")**

**plt.show()**

**from sklearn.model\_selection import GridSearchCV**

**parameters\_grid = {"n\_neighbors": np.arange(0,50)}**

**knn= KNeighborsClassifier()**

**knn\_GSV = GridSearchCV(knn, param\_grid=parameters\_grid, cv = 5)**

**knn\_GSV.fit(X, y)**

**from sklearn.model\_selection import GridSearchCV**

**from sklearn.neighbors import KNeighborsClassifier**

**import numpy as np**

**# Define the parameter grid**

**param\_grid = {'n\_neighbors': np.array(range(1, 50))} # Change to range for simplicity**

**# Create the GridSearchCV object**

**grid\_search = GridSearchCV(cv=5, estimator=KNeighborsClassifier(), param\_grid=param\_grid)**

**# Example data (X\_train, y\_train) should be defined here**

**# grid\_search.fit(X\_train, y\_train) # Uncomment this line when you have your data**

**# Now you can proceed with fitting the grid search**

**print("Best Params" ,knn\_GSV.best\_params\_)**

**print("Best score" ,knn\_GSV.best\_score\_)**