**Project healthcare-Capstone**

We download the "health care diabetes.csv" dataset provided and upload it into the Lab.

Read the Data - data=pd.read\_csv(‘health care diabetescsv')

**Printing the name of the columns -** data.columns

#### ****Printing the shape –** data.shape**

#### check Null values – data.isnull().sum()

#### #1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

#### #• Glucose

#### #• BloodPressure

#### #• SkinThickness

#### #• Insulin

#### #• BMI

#### #create histogram

#### plt.hist(data["Glucose"])

#### plt.show()

#### since 0 values are present, its not missing values, we can use replace function - we use median)

#### # use median to fill data value - as data is skewed

#### data['Glucose'].replace(0, data['Glucose'].median(),inplace=True)

#### #now missing values filled with median value

#### plt.hist(data["Glucose"])

#### plt.show()

#### # check blood pressure

#### plt.hist(data["BloodPressure"])

#### plt.show()

#### #now missing vlues filled with median value

#### plt.hist(data["BloodPressure"])

#### plt.show()

#### #create histogram

#### plt.hist(data["Insulin"])

#### plt.show()

#### #again missing values fill it with median

#### data['Insulin'].replace(0, data['Insulin'].median(),inplace=True)

#### #now missing vlues filled with median value

#### plt.hist(data["Insulin"])

#### plt.show()

#### #create histogram

#### plt.hist(data["SkinThickness"])

#### plt.show()

#### #again missing values fill it with median

#### data['SkinThickness'].replace(0, data['SkinThickness'].median(),inplace=True)

#### #now missing vlues filled with median value

#### plt.hist(data["SkinThickness"])

#### plt.show()

#### #create histogram

#### plt.hist(data["BMI"])

#### plt.show()

#### #again missing values fill it with median

#### data['BMI'].replace(0, data['BMI'].median(),inplace=True)

#### #now missing vlues filled with median value

#### plt.hist(data["BMI"])

#### plt.show()

#### #checking if any 0 value present

#### data[data['Insulin']==0]

#### #using for loop instead of individually solving & filling each variable.

#### # see below

#### var=['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','BMI']

#### #use for loop to treat missing values

#### for i in var:

#### data[i].replace(0,data[i].median(),inplace=True)

#### #create count plot - age is 25, show hs may along with plt we will show

1. 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.

#### #countplot is univariate analysis which canbe seen using histogram

#### plt.figure(figsize=(12,8))

#### sns.countplot(x=data['Age'])

#### plt.show()

#### plt.figure(figsize=(12,8))

#### sns.histplot(x=data['Glucose'])

#### plt.show()

#### plt.figure(figsize=(12,8))

#### sns.histplot(x=data['BloodPressure'])

#### plt.show()

#### data['Outcome'].value\_counts()

0 500

1 268

Name: Outcome, dtype: int64

#### #500 not having diabetes & vice versa

Project Task: Week 2 Data Exploration:

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

#### #count plot for target variable)

#### sns.countplot(x=data['Outcome'])

#### plt.show()

#### sns.countplot(x=data['Pregnancies'])

#### plt.show()

#### plt.figure(figsize=(12,8))

#### sns.countplot(x=data['DiabetesPedigreeFunction'])

#### plt.show()

#### plt.figure(figsize=(12,8))

#### sns.histplot(x=data['DiabetesPedigreeFunction'])

#### plt.show()

#### plt.figure(figsize=(12,8))

#### sns.countplot(x=data['BMI'])

#### plt.show()

#### plt.figure(figsize=(12,8))

#### sns.histplot(x=data['BMI'])

#### plt.show()

#### plt.figure(figsize=(12,8))

#### sns.countplot(x=data['Glucose'])

#### plt.show()

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

#### sns.scatterplot(x=data['BloodPressure'],y=data['Glucose'])

#### plt.show()

#### #same plot with outcome variable

#### #high glucose is havin diabetes, high bloodpressure as well.glucose level is associated with Outcome

#### #glucose visually more associated than blood pressure

#### sns.scatterplot(x=data['BloodPressure'],y=data['Glucose'], hue='Outcome',data=data)

#### plt.show()

#### #how to check if glucose if affecting more or blood pressure the outcome,

#### #we can do univariate anlysis to find which is more corelated or do correalation analysis

#### data.corr()

#### plt.figure(figsize=(12,8))

#### sns.heatmap(data.corr(),annot=True)

#### #from th gaph we can see outcome variable in last line, glucose affects outcome more, Outcome variable is highlly corelated with glucose, next is bmi which affects

#### sns.scatterplot(x=data['BMI'],y=data['Glucose'], hue='Outcome',data=data)

#### plt.show()

#### sns.scatterplot(x=data['Insulin'],y=data['Glucose'], hue='Outcome',data=data)

#### plt.show()

#### sns.scatterplot(x=data['Insulin'],y=data['BMI'], hue='Outcome',data=data)

#### plt.show()

Project Task: Week 3 Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.
2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.

#### #1. extract indep & dep var

#### #2. convert cat into numerical data

#### #3. Create train & test split

#### #4. apply scaling to normalise data

#### #5. apply ML algorithm

#### #6. evaluate /Predict Model

#### #iloc method, -1 is becaus elast value is our target varaible& .values is to convert ot to to array, collon is used for all columns

#### X=data.iloc[:,:-1].values #input features

#### y=data.iloc[:,-1].values #target features

#### # TRAIN SET TEST SET SPLIT

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

#### X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2, random\_state=42)

#### X\_train.shape

#### X\_test.shape

#### X\_train

#### #apply first loggistic regression

#### from sklearn.linear\_model import LogisticRegression

#### classifier\_log\_reg=LogisticRegression()

#### import warnings

#### warnings.filterwarnings('ignore')

#### #prediction

#### y\_pred=classifier\_log\_reg.predict(X\_test)

#### #evaluate ML model

#### from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report

#### print(confusion\_matrix(y\_test,y\_pred))

#### print('Test accuracy Score',accuracy\_score(y\_test,y\_pred))

#### y\_pred\_train=classifier\_log\_reg.predict(X\_train)

#### #to check overfiting

#### print('Train accuracy Score',accuracy\_score(y\_train,y\_pred\_train))

#### #print classification report to see where accuracy & recall is down

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

#### #prepare AUC& ROC Curve

#### from sklearn.metrics import roc\_auc\_score,roc\_curve

#### prob=classifier\_log\_reg.predict\_proba(X\_test)

#### prob\_pos=prob[:,1]

#### auc\_score=roc\_auc\_score(y\_test,prob\_pos)

#### print("AUC Score:",auc\_score)

#### #prob= probablilty = array

#### prob

#### #LHS ones are probbility of 0 class & RHS are for 1 class, from this prob arraye, we use sigmoid function to predict probabilities for output clas in logistic regression

#### #thrs are thresholds

#### fpr, tpr,thrs=roc\_curve(y\_test,prob\_pos)

#### plt.plot([0,1],[0,1],linestyle='--')

#### plt.plot(fpr,tpr,marker='.')

#### plt.show()

#### # save the model

#### import joblib

#### joblib.dump(classifier\_log\_reg,'Log\_Reg.pk1')

#### print('model is saved')

#### #apply first loggistic regression

#### from sklearn.ensemble import RandomForestClassifier

#### #training

#### classifier\_rf.fit(X\_train,y\_train)

#### classifier\_rf=RandomForestClassifier()

#### #prediction

#### y\_pred1=classifier\_rf.predict(X\_test)

#### print('Test rf Score',accuracy\_score(y\_test,y\_pred1))

#### #apply first loggistic regression

#### from sklearn.neighbors import KNeighborsClassifier

#### classifier\_knn=KNeighborsClassifier()

#### #training

#### classifier\_knn.fit(X\_train,y\_train)

#### #prediction

#### y\_pred2=classifier\_knn.predict(X\_test)

#### print('Test knn Score',accuracy\_score(y\_test,y\_pred2))