Cell No.:1

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn.externals import joblib

**Data Description:**

Cell No.:2

## load the data

diabetesDF = pd.read\_csv('diabetes.csv')

print(diabetesDF.head())

**Data Exploration:**

Cell No.:3

corr = diabetesDF.corr()

print(corr)

**Dataset Preparation:**

Cell No.:4

# split into train and test

dfTrain = diabetesDF[:650]

dfTest = diabetesDF[650:750]

dfCheck = diabetesDF[750:]

Cell No.:5

# split features from target variable

trainLabel = np.asarray(dfTrain['Outcome'])

trainData = np.asarray(dfTrain.drop('Outcome',1))

testLabel = np.asarray(dfTest['Outcome'])

testData = np.asarray(dfTest.drop('Outcome',1))

Cell No.:6

# normalize the data

# makes it easier to interpret the model by looking at its weights

means = np.mean(trainData, axis=0)

stds = np.std(trainData, axis=0)

trainData = (trainData - means)/stds

testData = (testData - means)/stds

**ML Model**

Cell No.:7

# models target t as sigmoid(w0 + w1\*x1 + w2\*x2 + ... + wd\*xd)

diabetesCheck = LogisticRegression()

diabetesCheck.fit(trainData, trainLabel)

accuracy = diabetesCheck.score(testData, testLabel)

print("accuracy = ", accuracy \* 100, "%")

**Checking ML model**

Cell No.:8

coeff = list(diabetesCheck.coef\_[0])

labels = list(dfTrain.drop('Outcome',1).columns)

features = pd.DataFrame()

features['Features'] = labels

features['importance'] = coeff

features.sort\_values(by=['importance'], ascending=True, inplace=True)

features['positive'] = features['importance'] > 0

features.set\_index('Features', inplace=True)

features.importance.plot(kind='barh', figsize=(11, 6),color = features.positive.map({True: 'blue', False: 'red'}))

plt.xlabel('Importance')

Cell No.:9

joblib.dump([diabetesCheck, means, stds], 'diabeteseModel.pkl')

## load model

diabetesLoadedModel, means, stds = joblib.load('diabeteseModel.pkl')

accuracyModel = diabetesLoadedModel.score(testData, testLabel)

print("accuracy = ",accuracyModel \* 100,"%")

**Predicting Using Model**

Cell No.:10

## making predictions

sampleData = dfCheck[:1]

# prepare sample

sampleDataFeatures = np.asarray(sampleData.drop('Outcome',1))

sampleDataFeatures = (sampleDataFeatures - means)/stds

# predict

predictionProbability = diabetesCheck.predict\_proba(sampleDataFeatures)

prediction = diabetesCheck.predict(sampleDataFeatures)

print('Probability:', predictionProbability)

print('prediction:', prediction)

// Dataset: https://www.kaggle.com/uciml/pima-indians-diabetes-database

//Thanks Anant for the reference.