DDA_Formative_Solutions

April 1, 2022

1. In this notebook let us use KNN to build a machine learning model using k-Nearest Neighbors algorithm to predict whether the patients in the "Pima Indians Diabetes Dataset" have diabetes or not.

[1]:
```
# Run this cell to load the data from a URL

data ='https://storage.googleapis.com/kagglesdsdata/datasets/228/482/diabetes.
 ↪csv?
 ↪X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.
 ↪iam.gserviceaccount.
 ↪goog%4_request&X-Goog-Date=20211201T124634Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&
```

[10]:
```
# Import the data in Python



# Provide your solution here.
import pandas as pd

columns = ['Preg', 'Glu', 'BP', 'ST',
        'Insulin', 'BMI', 'DPF', 'Age', 'Outcome']

diabetes_data = pd.read_csv('pima-indians-diabetes.csv', header=None)
diabetes_data.columns = columns
```

2. Explore the data by printing the first 5 records.

[11]:
```
# Provide your solution here.

diabetes_data.head(5)
```

[11]:

	Preg	Glu	BP	ST	Insulin	BMI	DPF	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

3. Provide a short paragraph (100-200 words to discuss eht pima dataset.

[16]:
```
# Provide your solution here.

# The Pima dataset comes in a structured format in the form of a csv file
#
```

The pima dataset comes in a structured format in the form of a csv file.
It has 9 columns and 768 rows.
It describes the medical record for pima indians

`Fields description follow:`

preg = Number of times pregnant

plas = Plasma glucose concentration a 2 hours in an oral glucose tolerance test

pres = Diastolic blood pressure (mm Hg)

skin = Triceps skin fold thickness (mm)

test = 2-Hour serum insulin (mu U/ml)

mass = Body mass index (weight in kg/(height in m)^2)

pedi = Diabetes pedigree function

age = Age (years)

class = Class variable (1:tested positive for diabetes, 0: tested negative for diabetes)

[18]:
```
diabetes_data.describe()
```

[18]:

	Preg	Glu	BP	ST	Insulin	BMI
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000

	DPF	Age	Outcome
count	768.000000	768.000000	768.000000
mean	0.471876	33.240885	0.348958
std	0.331329	11.760232	0.476951
min	0.078000	21.000000	0.000000
25%	0.243750	24.000000	0.000000
50%	0.372500	29.000000	0.000000
75%	0.626250	41.000000	1.000000
max	2.420000	81.000000	1.000000

4. Print the shape of the dataset

[13]:
```
# Provide your solution here.

diabetes_data.shape
```

[13]:
```
(768, 9)
```

5. Complete the following.

[15]:
```
# Number of rows:
# Number of columns:

print("Number of rows:", diabetes_data.shape[0])
print("Number of columns:", diabetes_data.shape[1])
```

```
Number of rows: 768
Number of columns: 9
```

6. What can you use as features? and what can you use as a target variable?

[ ]:
```
# Provide your solution here.

# Age and preg can be used as features for example

# Outcome could be a target variable
```

7. Do we have any empty records? Extract the empty row counts per column.

[19]:
```
# Provide your solution here.

diabetes_data.isna().sum()
```

[19]:
```
Preg       0
Glu        0
BP         0
ST         0
Insulin    0
BMI        0
DPF        0
Age        0
Outcome    0
dtype: int64
```

8. Provide a script to count how many data cells there are in the dataframe for each caterogy (group by Outcome).

[22]:
```
# Provide your solution here.

diabetes_data.groupby('Outcome').size()
```

[22]:
```
Outcome
0    500
1    268
dtype: int64
```

[ ]:
```
9. We will need to split our dataset into feature data (X) and target data (y).

As features, we can use the next:
*  Glucose
*  BloodPressure
*  Insulin
*  BMI

(Please note that this is a random selection of features, you can include more␣
 ↪columns if you like)

As targets, we can use the next:
* Outcome

Extract the data for the selected four columns and store it in a new dataframe␣
 ↪called *feature_data*.

Print the first 5 records of the feature data.
```

[44]:
```
# Provide your solution here

feature_df = diabetes_data[['Glu', 'BP', 'Insulin', 'BMI']]
print(feature_df.head(5))
```

```
   Glu  BP  Insulin   BMI
0  148  72        0  33.6
1   85  66        0  26.6
2  183  64        0  23.3
3   89  66       94  28.1
4  137  40      168  43.1
```

10. Print the first 5 records of the feature data.Then create the *target_data* using the Outcome column.

Print the first 5 records of the target data.

[43]:
```
# Provide your solution here.

target_df = diabetes_data[['Outcome']]

target_df
type(target_df)
```

[43]:
```
pandas.core.frame.DataFrame
```

11. How many feature and how many target data do we have?

[36]:
```
# Provide your solution here.
```

12. Let's create our **X** and **y** variables to use it in our classification models.

[70]:
```
# Provide your solution here.
X=feature_df
y=target_df
```

13. Create your KNN model and fit it using the X and y variables from the previous step. Use the *KNeighborsClassifier* including 4 neighbors.

[84]:
```
# Provide your solution here.

from sklearn.neighbors import KNeighborsClassifier
import numpy as np

knn = KNeighborsClassifier(n_neighbors=4)

#https://stackoverflow.com/questions/34165731/
 ↪a-column-vector-y-was-passed-when-a-1d-array-was-expected
# np ravel used as per link above. was receiving error without it
knn.fit(X.values,np.ravel(y))
```

[84]:
```
KNeighborsClassifier(n_neighbors=4)
```

14. Print the first five records, we will use record 0 to explore the prediction of our model.

[86]:
```
# Provide your solution here.

print(diabetes_data.head(5))
```

```
   Preg  Glu  BP  ST  Insulin   BMI    DPF  Age  Outcome
0     6  148  72  35        0  33.6  0.627   50        1
1     1   85  66  29        0  26.6  0.351   31        0
2     8  183  64   0        0  23.3  0.672   32        1
3     1   89  66  23       94  28.1  0.167   21        0
4     0  137  40  35      168  43.1  2.288   33        1
```

15. Let's use record: 148,72,0,33.6 to print the model prediction.

[87]:
```
# Provide your solution here.

knn.predict([[148,72,0,33.6]])

#print(feature_df.iloc[0])
```

[87]:
```
array([0], dtype=int64)
```

16. Was the prediction correct? What do you think is the problem?

[ ]:
```
# It was incorrect the model predicted 0 (tested negative for diabetes)
# I believe this is because of using a smaller number of feature variables to␣
 ↪train our model
# Increasing the number of feature variables may have helped in the accuracy of␣
 ↪the prediction
```

17. Create a new *LogisticRegression model* and then fit the **X** and **y** dataframes that we just created. You can set the max_iter=200.

[117]:
```
# Provide your solution here.
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression(max_iter=200)
reg.fit(X.values, np.ravel(y))

y_predict = reg.predict([[148,72,0,33.6]])
#dataframes = [feature_df, target_df]
#LogisticRegression = pd.concat(dataframes)

print(y_predict)
```

```
[1]
```

16. Was the prediction correct?

[ ]:
```
# Provide your solution here.
#The prediction using the same row was correct and as expected
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  "X does not have valid feature names, but"
```

[ ]:
```
array([1])
```

17. Can you plot a pairplot for the following features:

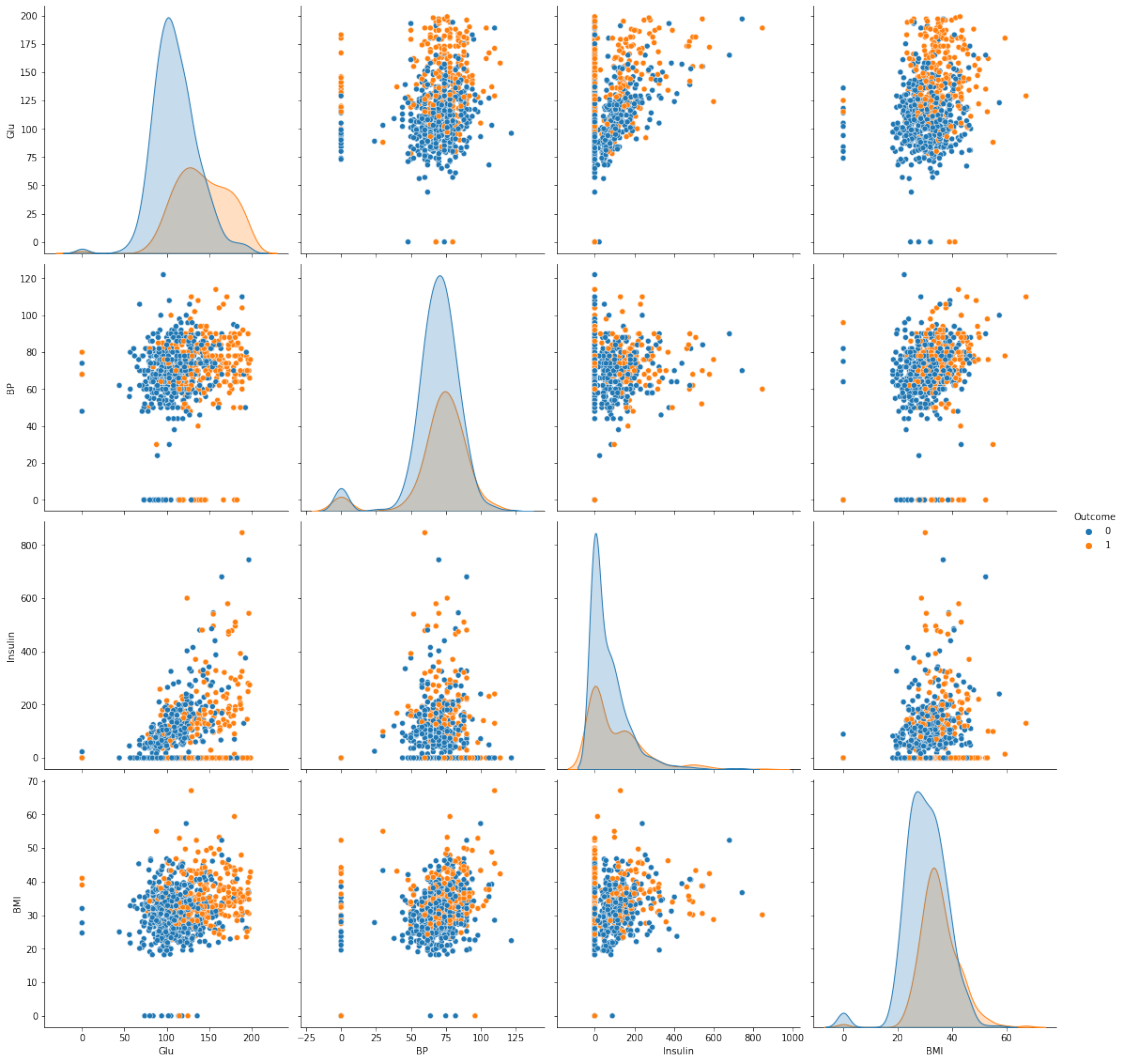
- 'Glucose','BloodPressure','Insulin','BMI','Outcome'

[106]:
```
# Provide your solution here.

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.figure()
sns.pairplot(diabetes_data[['Glu', 'BP', 'Insulin', 'BMI', 'Outcome']],␣
 ↪hue='Outcome', height=4 )
```

```
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

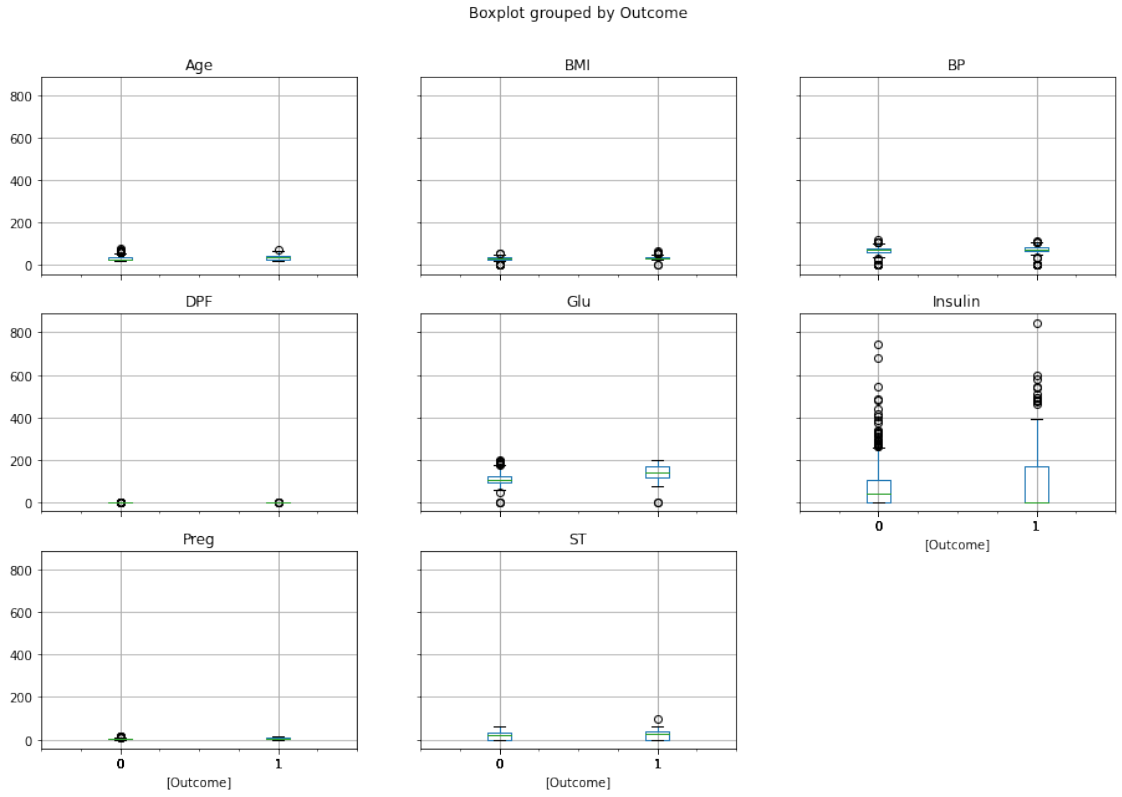


18. Plot a boxplor for the *data* by *Outcome*. This will plot diagrams for each feature and the associated confidence intervals per outcome (0 or 1).

[101]:
```
# Provide your solution here.


plt.figure()
diabetes_data.boxplot(by='Outcome', figsize=(15,10))
plt.show()
```

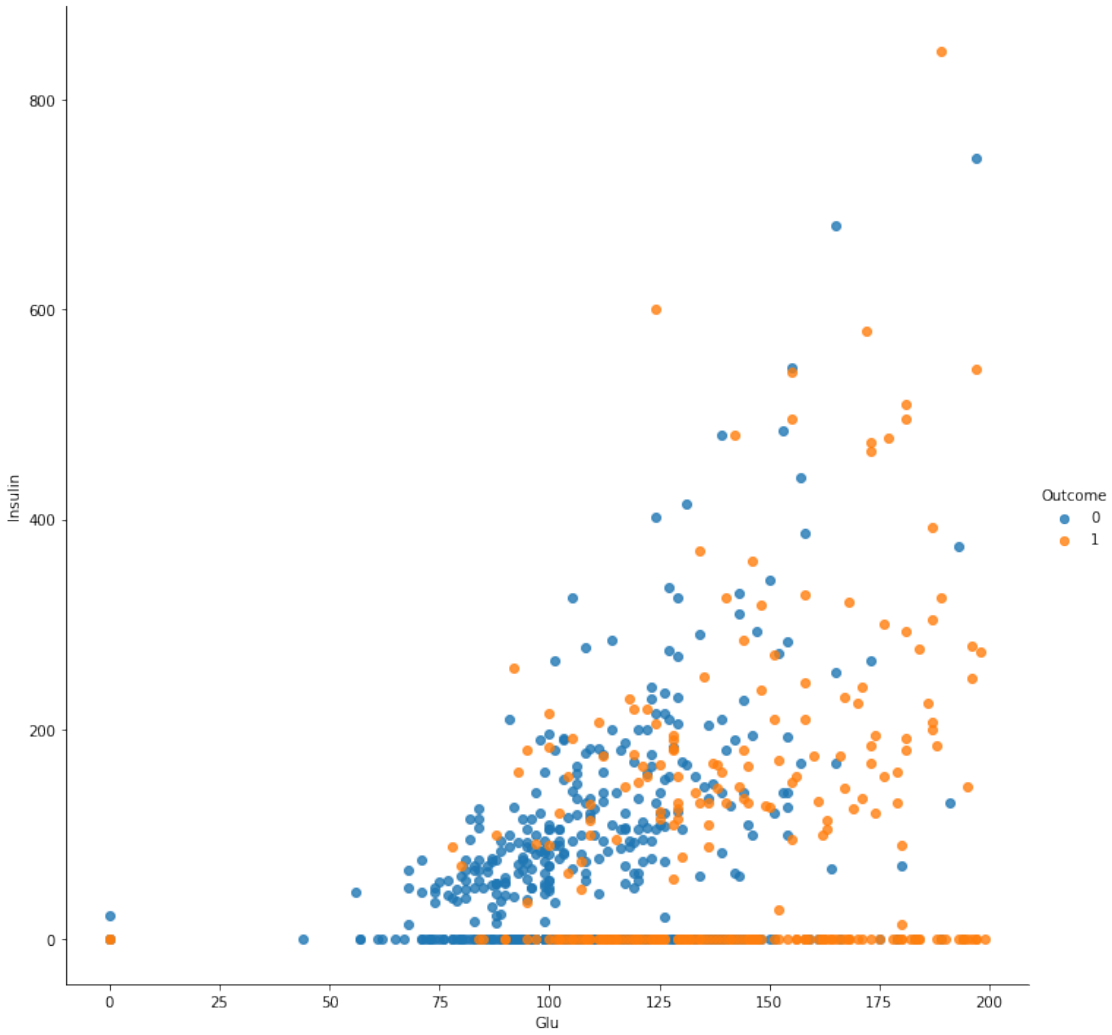
```
<Figure size 432x288 with 0 Axes>
```



19. Plot a diagram to show a scatterplot of the *data* using x as *Glucose* and as y the *Insulin*. Use the *Outcome* as hue.

[108]:
```
# Provide your solution here.
sns.lmplot(x='Glu', y='Insulin', hue='Outcome', data=diabetes_data, height=10,␣
↪fit_reg=False)
```

[108]:
```
<seaborn.axisgrid.FacetGrid at 0x2337d1c6d70>
```

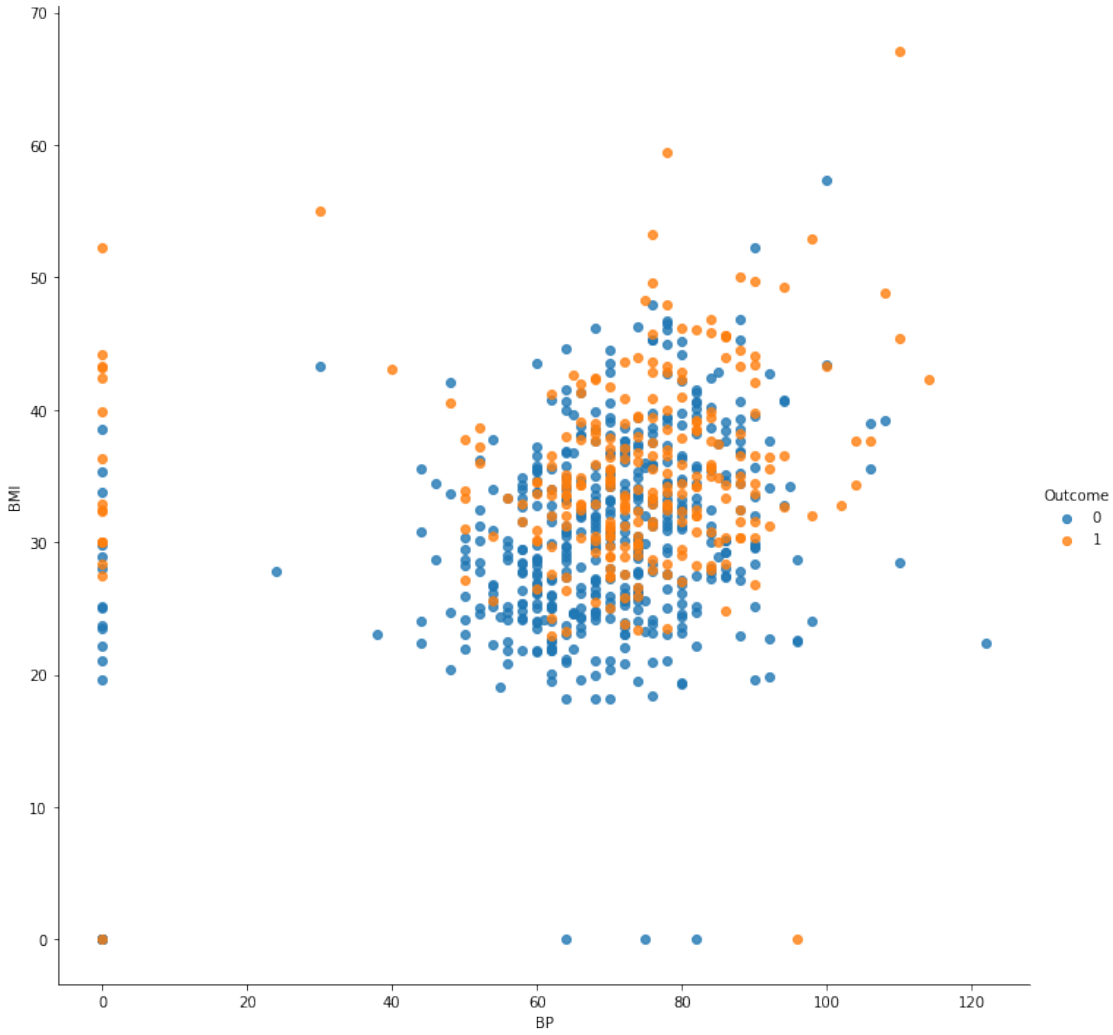


20. Provide a scatterplot to find the relationship between Blood Pressure and BMI.

[110]:
```
# Provide your solution here.

sns.lmplot(x='BP', y='BMI', hue='Outcome', data = diabetes_data, height=10,␣
↪fit_reg=False)
```

[110]: <seaborn.axisgrid.FacetGrid at 0x2337d4abac0>


70
60
50
40
30
20
10
0
BMI
0
20
40
60
80
100
120
BP
Outcome
0
1

[ ]: