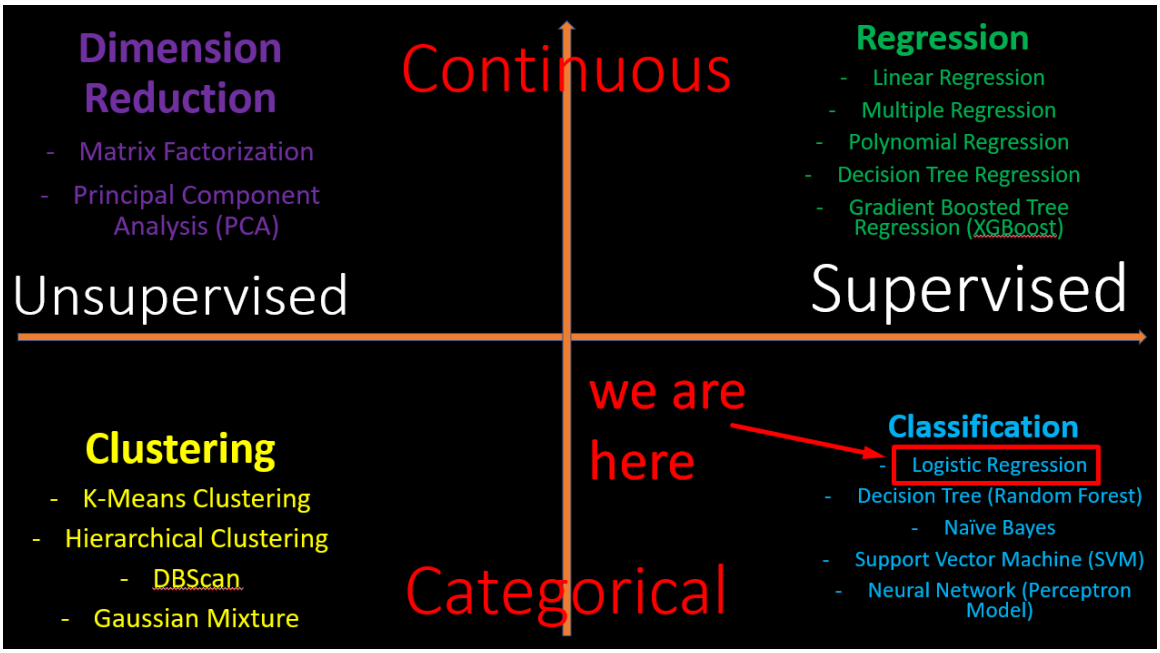
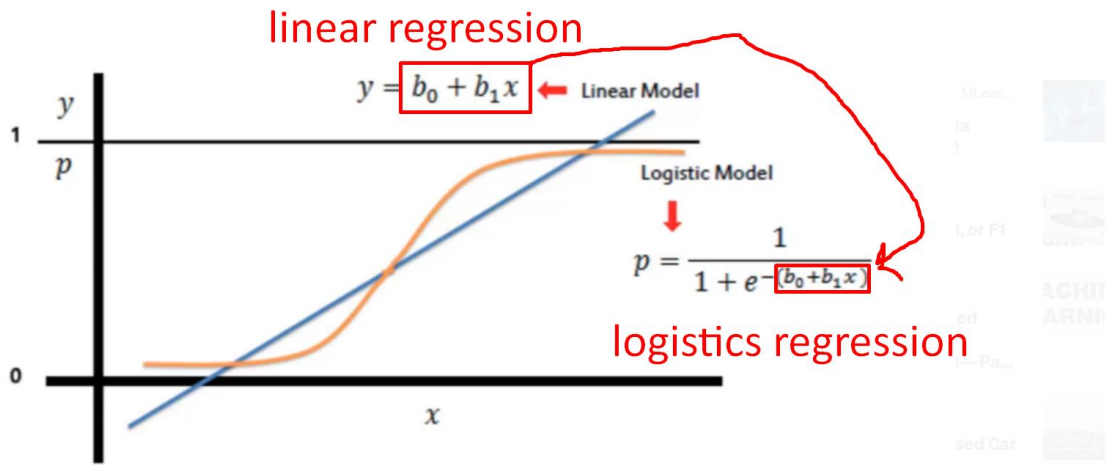
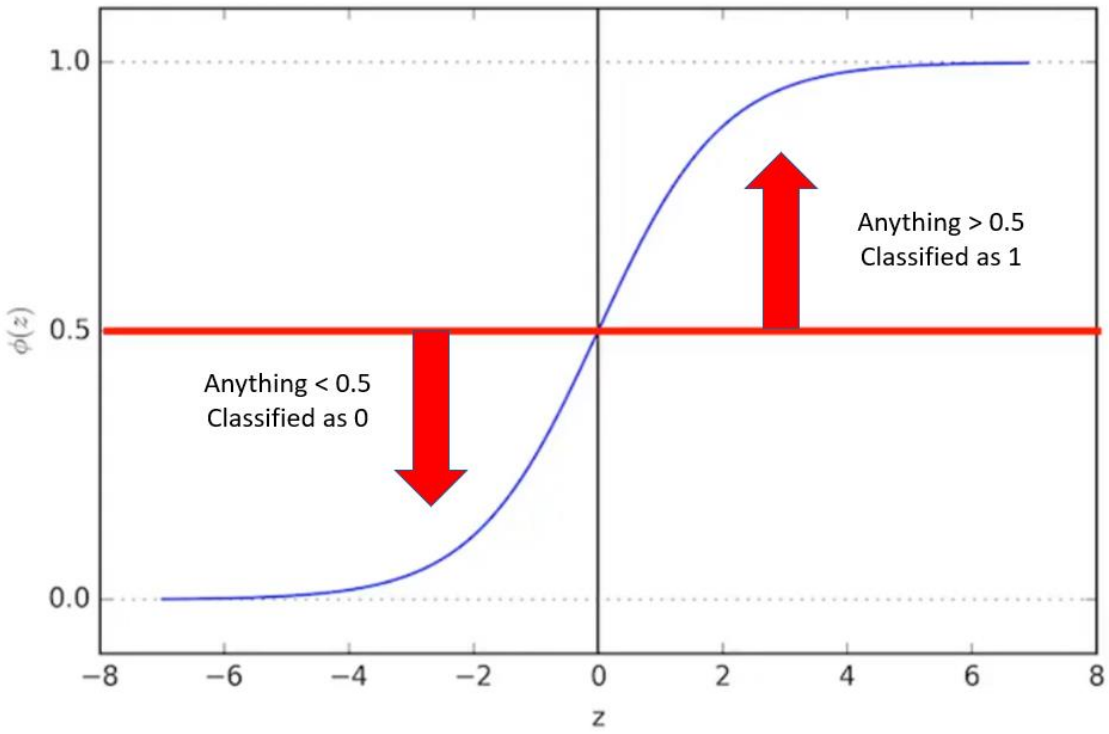
LOGISTICS REGRESSION

USING PYTHON
DR. ALVIN ANG

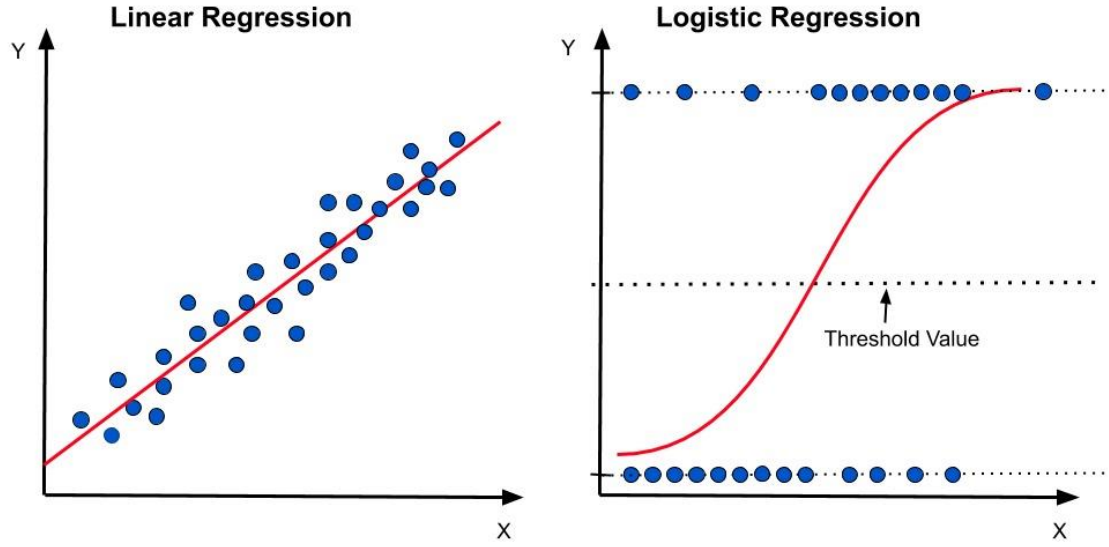
CONTENTS







THRESHOLD VALUE = 0.5



II. STUDENT STUDY HOURS EXAMPLE (1 PREDICTOR VARIABLE)

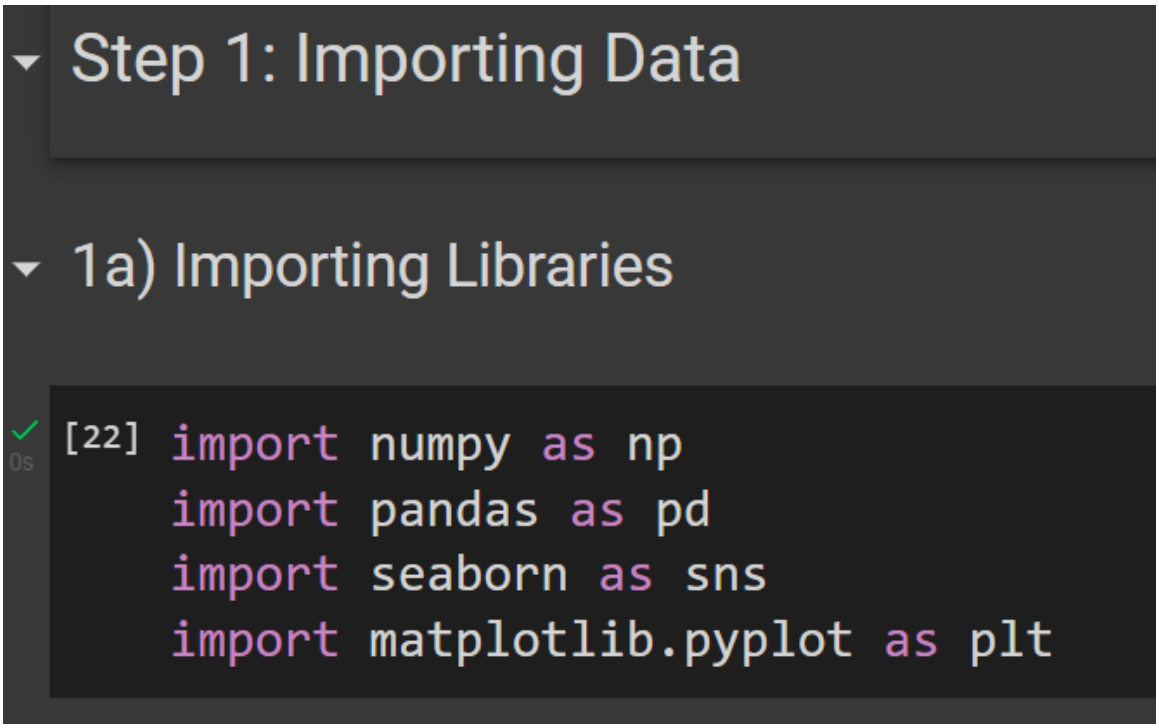
https://www.alvinang.sg/s/Logistics_Regression_with_Python_Student_Study_Hours_Example_by_Dr_Alvin_Ang.ipynb

https://www.alvinang.sg/s/results.csv

https://towardsdatascience.com/logistic-regression-in-python-2f965c355b93

A. STEP 1: IMPORTING DATA

1. IMPORTING LIBRARIES



2. IMPORTING DATASET

1b) Importing Dataset

```
[ ] df=pd.read_csv("https://www.alvinang.sg/s/results.csv")
    df

    #It appears that the longer hours a student put into studying,
    #his results gets better and better.
```

	Hours	StudentId	Result
0	1.0	10	Fail
1	1.2	15	Fail
2	1.5	21	Fail
3	1.7	16	Fail
4	2.0	14	Fail
5	2.4	5	Pass
6	2.5	7	Fail
7	2.6	2	Pass
8	2.8	17	Fail
9	3.0	18	Pass
10	3.5	6	Pass
11	3.7	20	Pass
12	3.9	3	Pass
13	4.0	13	Pass
14	4.5	12	Pass
15	5.0	19	Pass
16	3.1	24	Fail
17	0.5	1	Fail

3. MAPPING FAIL TO 0 AND PASS TO 1

1c) Mapping Fail to 0 and Pass to 1

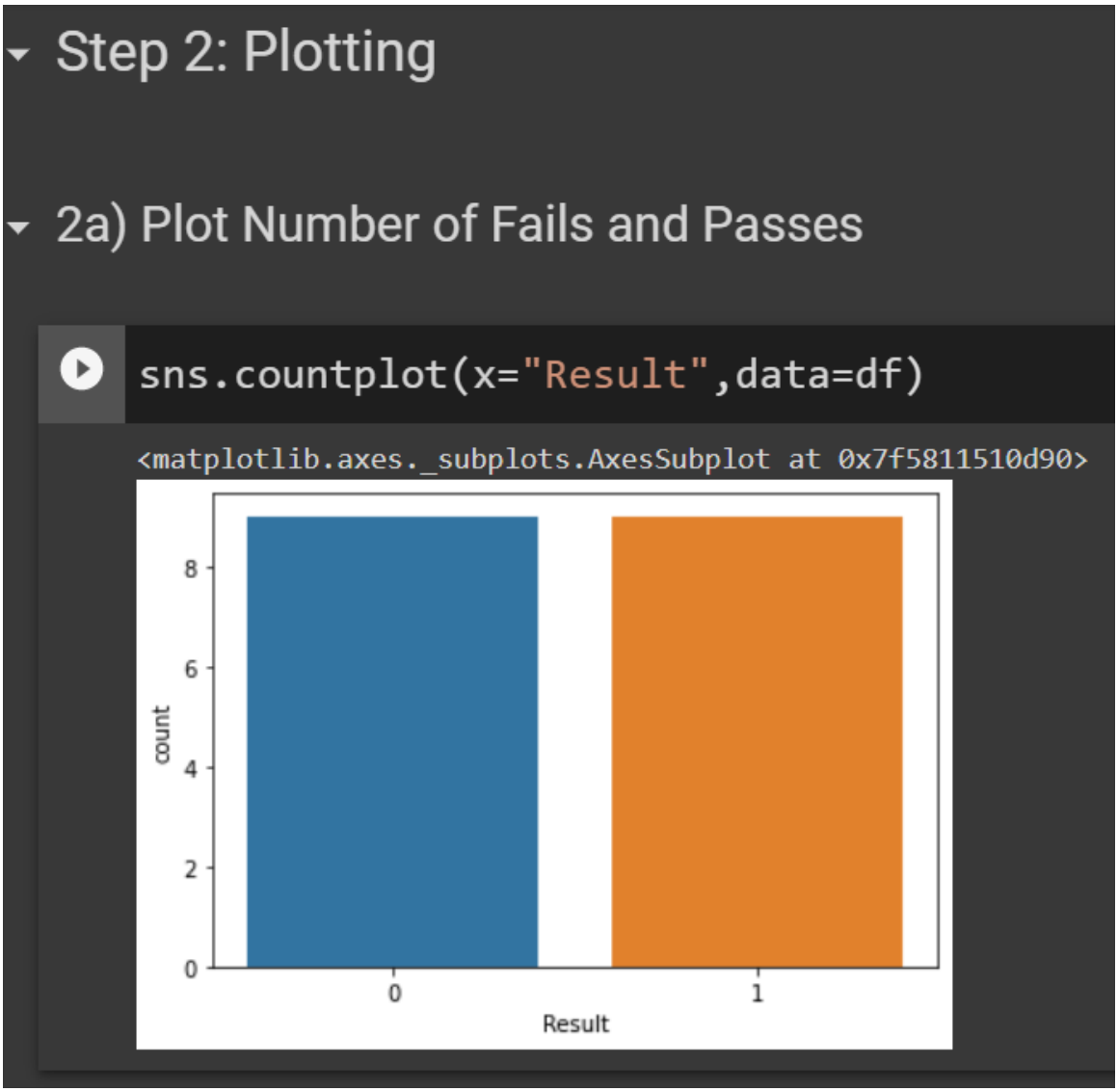
```
[ ] df['Result'] = df['Result'].map({'Fail':0, 'Pass':1})
```

```
df
```

	Hours	StudentId	Result
0	1.0	10	0
1	1.2	15	0
2	1.5	21	0
3	1.7	16	0
4	2.0	14	0
5	2.4	5	1
6	2.5	7	0
7	2.6	2	1
8	2.8	17	0
9	3.0	18	1
10	3.5	6	1
11	3.7	20	1
12	3.9	3	1
13	4.0	13	1
14	4.5	12	1
15	5.0	19	1
16	3.1	24	0
17	0.5	1	0

B. STEP 2: PLOTTING

1. PLOT NUMBER OF FAILS AND PASSES

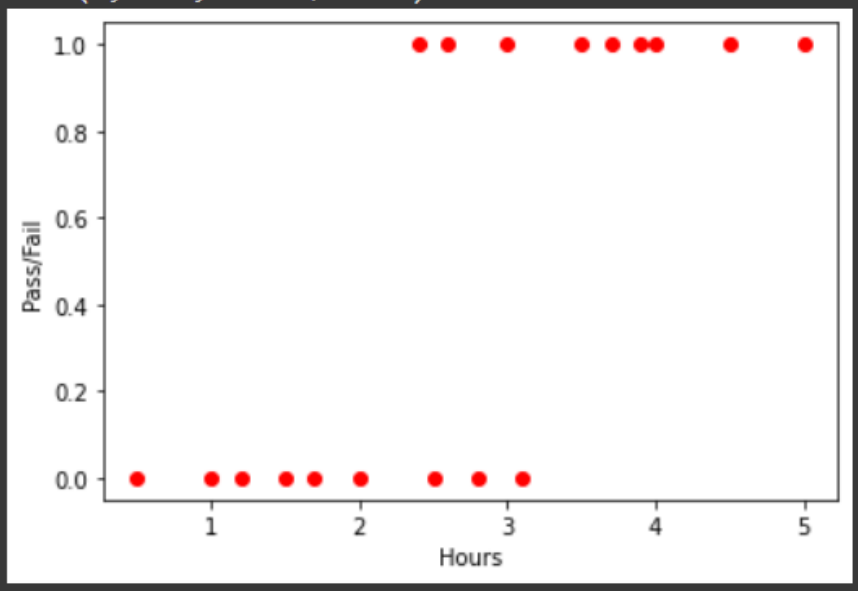


2. PLOT PASS / FAIL VS NUMBER OF HOURS STUDIED

2b) Plot Pass/Fail vs Number of Hours Studied

```
[ ] plt.scatter(df.Hours,df.Result,color='red')
    plt.xlabel("Hours")
    plt.ylabel("Pass/Fail")
```

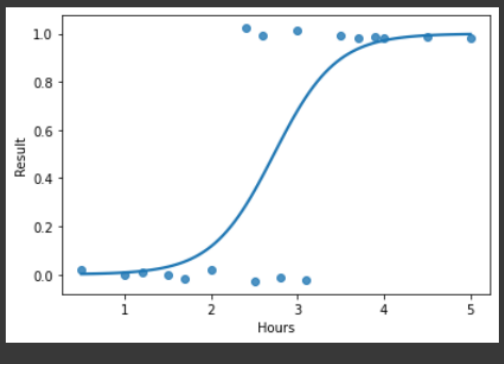
```
Text(0, 0.5, 'Pass/Fail')
```



3. PLOT LOGISTICS REGRESSION CURVE

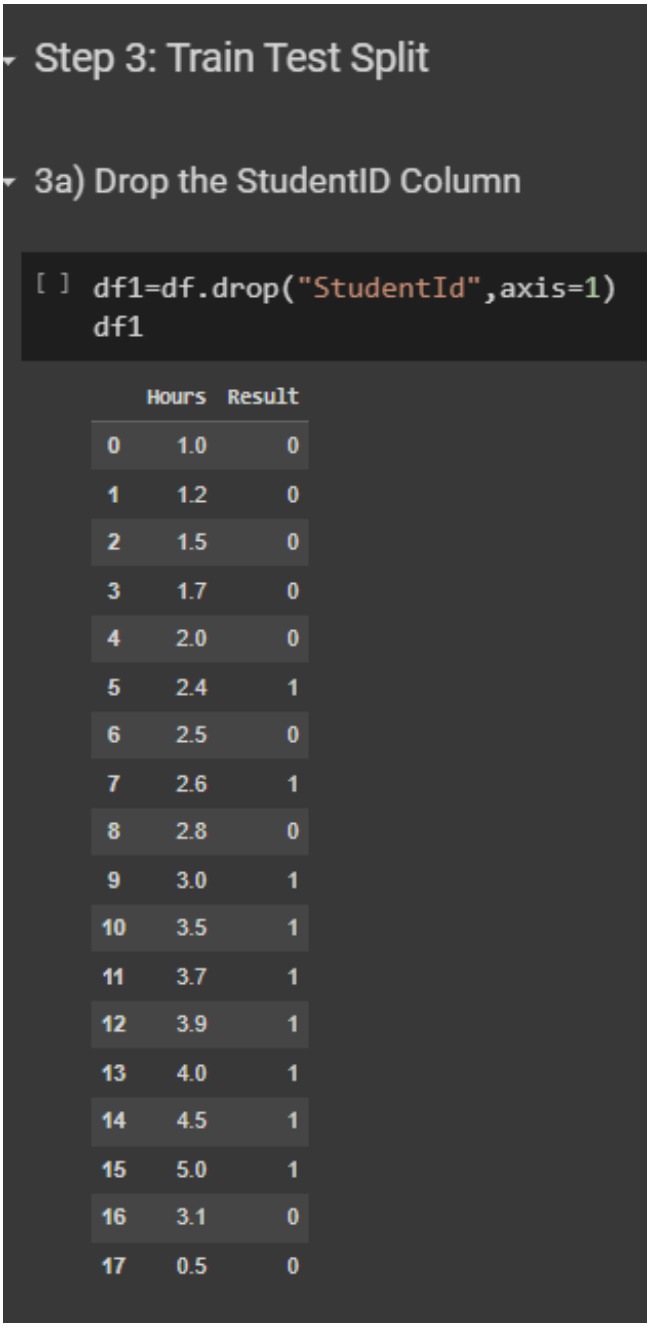
2c) Plot Logistics Regression Curve

```
sns.regplot(x = df.Hours, y = df.Result, y_jitter=0.03, data=df, logistic = True, ci = None)

plt.show()

#the Threshold Value of 0.5 (point between 0 and 1) seems to appear at 3 hours
#meaning, anything > 3 hours --> PASS
#anything < 3 hours --> FAIL
```



C. STEP 3: TRAIN TEST SPLIT

1. DROP THE STUDENTID COLUMN



Step 3: Train Test Split

3a) Drop the StudentID Column

```
[ ] df1=df.drop("StudentId",axis=1)
    df1
```

	Hours	Result
0	1.0	0
1	1.2	0
2	1.5	0
3	1.7	0
4	2.0	0
5	2.4	1
6	2.5	0
7	2.6	1
8	2.8	0
9	3.0	1
10	3.5	1
11	3.7	1
12	3.9	1
13	4.0	1
14	4.5	1
15	5.0	1
16	3.1	0
17	0.5	0

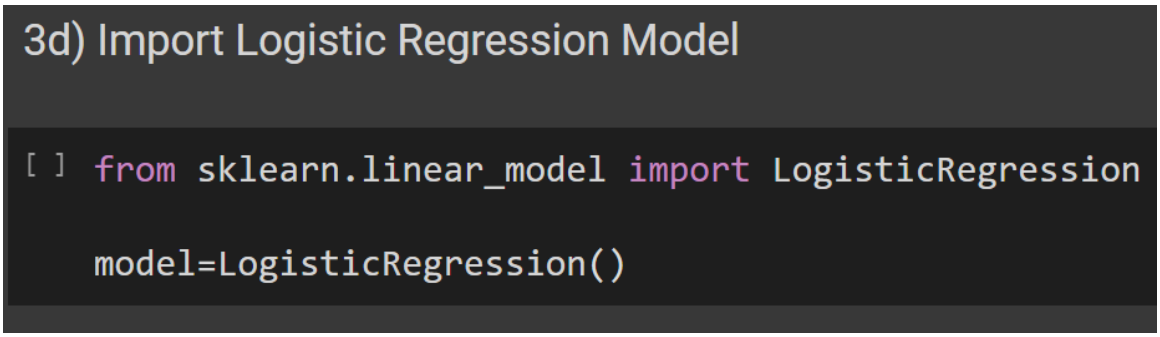
2. SEPARATE INTO 'X: HOURS' VS 'Y: RESULT'

```
3b) Seperate into 'x: Hours' vs 'y: Result'

[ ] x = df1.iloc[:,0:1]
    y = df1.iloc[:,1:]

    # x--> Hours
    # y --> Result
```

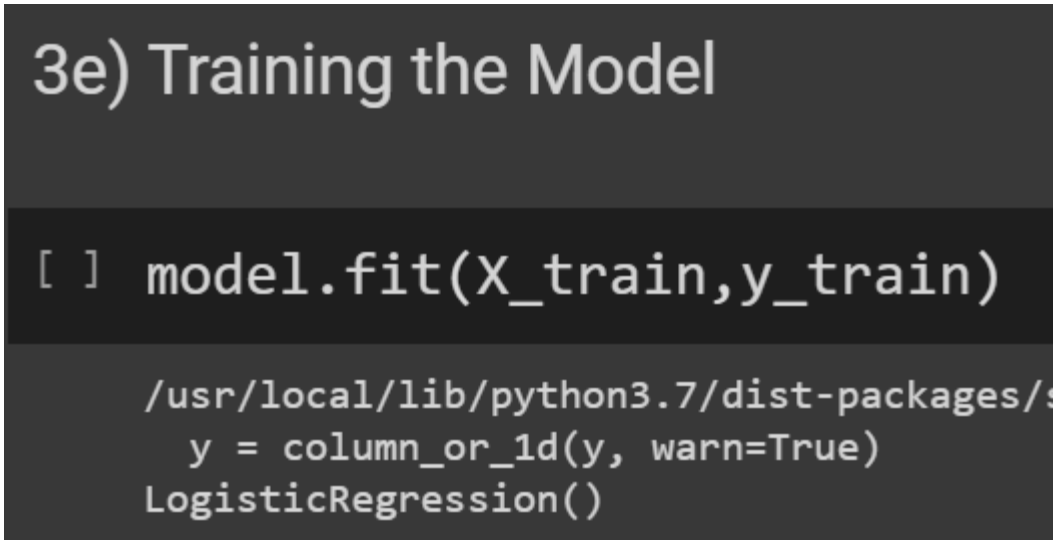
3. TRAIN TEST SPLIT

```
3c) Train Test 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=2)

#20% for testing, 80% for training
```

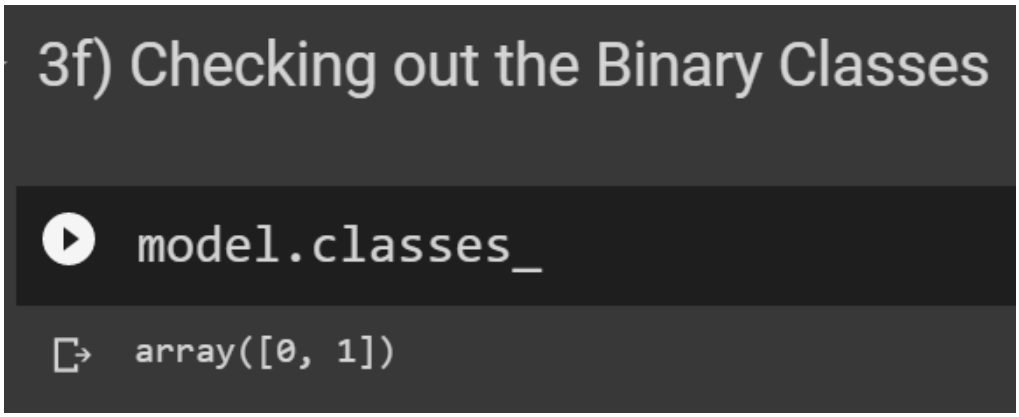
4. IMPORT LOGISTIC REGRESSION MODEL



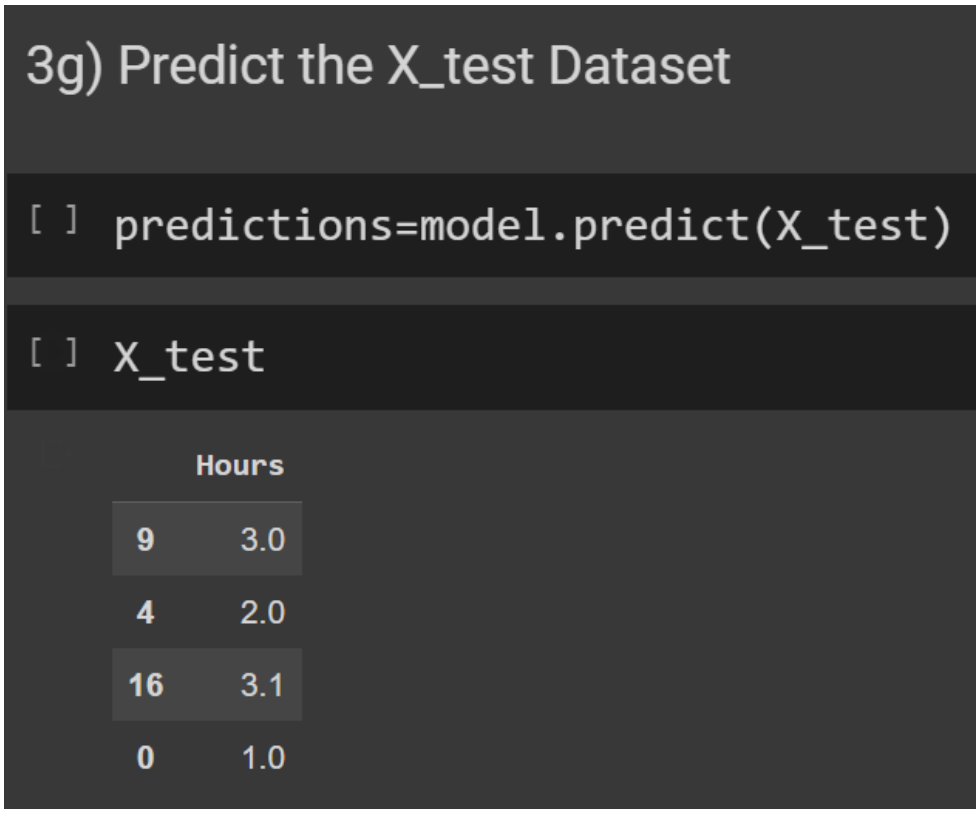
5. TRAINING THE MODEL



6. CHECKING OUT THE BINARY CLASSES



7. PREDICT THE X_TEST DATASET



```
predictions

#meaning, study for 3 hours --> Pass
#study for 2 hours --> Fail
#study for 3.1 hours --> Pass
#Study for 1 hour --> Fail
```

```
array([1, 0, 1, 0])
```

```
probabilities = model.predict_proba(X_test)

print(probabilities)

#66% chance that if you study 3 hours you will pass
#67% chance that if you study 2 hours you will fail
#69% chance that if you study 3.1 hours you will pass
#89% chance that if you study 1 hour you will fail
```

```
[[0.33864801 0.66135199]
 [0.67677464 0.32322536]
 [0.30785719 0.69214281]
 [0.8954164  0.1045836 ]]
```

D. STEP 4: METRICS

1. CONFUSION MATRIX

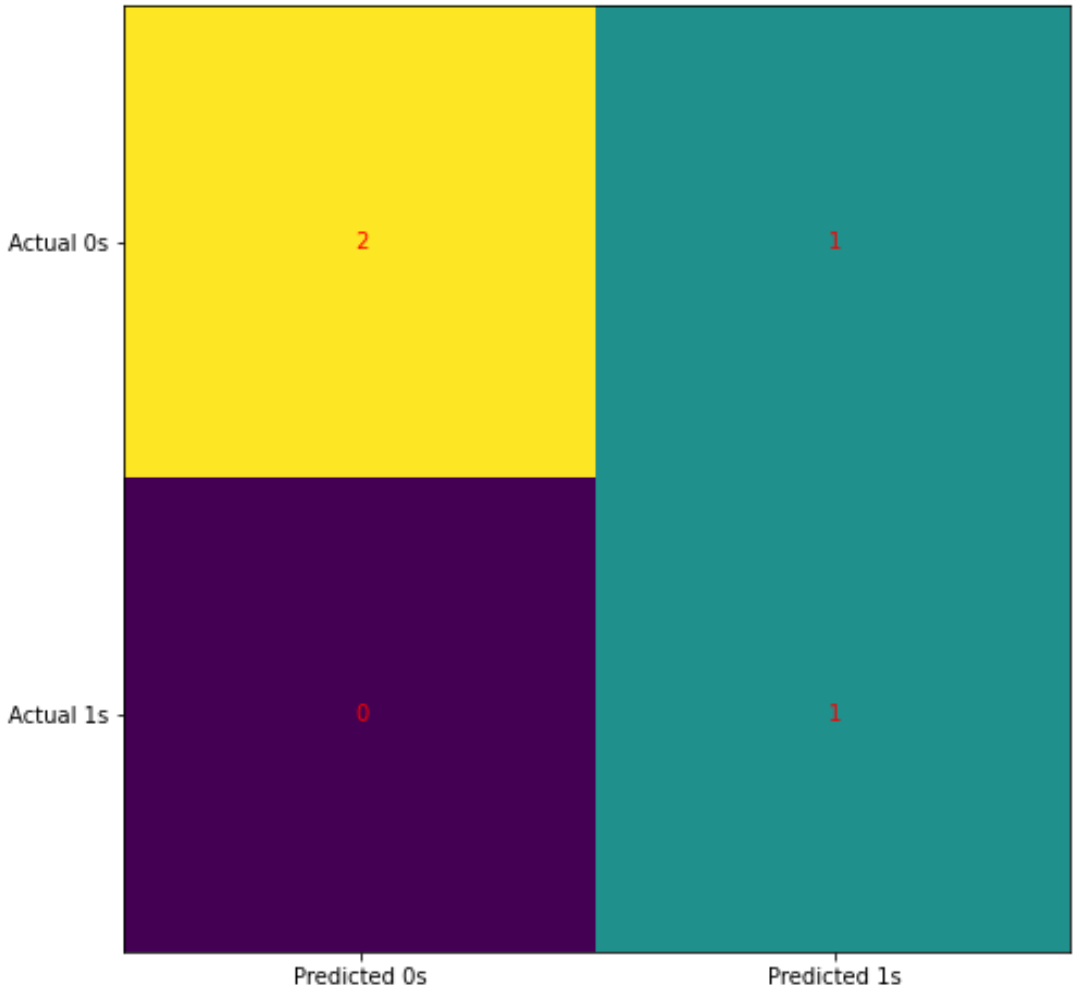
Step 4: Metrics

4a) Confusion Matrix

```
[ ] from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y_test,predictions)
    print (cm)
```

```
[[2 1]
 [0 1]]
```

```
fig, ax=plt.subplots(figsize = (8,8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1),ticklabels=('Predicted 0s','Predicted 1s'))
ax.yaxis.set(ticks=(0,1),ticklabels=('Actual 0s','Actual 1s'))
ax.set_ylim(1.5,-0.5)

for i in range (2):
  for j in range (2):
    ax.text(j,i,cm[i,j],ha='center',va='center',color='red')

plt.show()

#we have ONE Type 1 Error occuring: Actual FAIL but Predicted PASS
```



2. ACCURACY SCORE

4b) Accuracy Score

```
[ ] from sklearn.metrics import accuracy_score

    accuracy_score(y_test,predictions)

    #1/4 of the actual y_test labeled was predicted wrongly --> 25% reduction in accuracy
    #thus accuracy = 25%
```

```
0.75
```

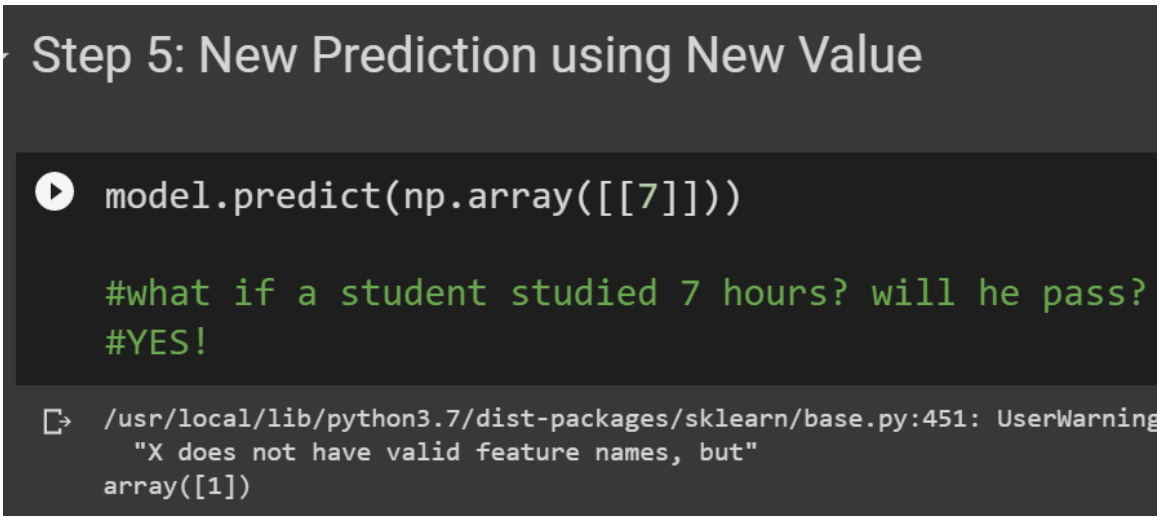
3. CLASSIFICATION REPORT

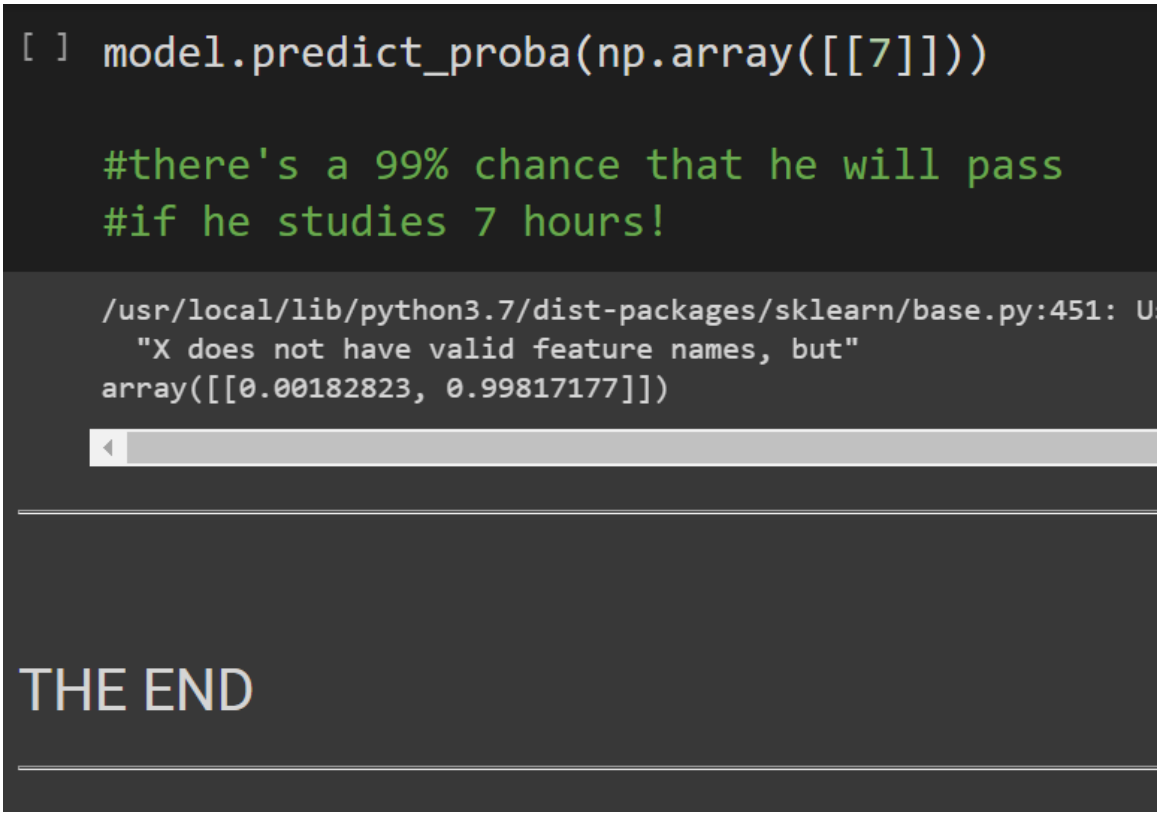
4c) Classification Report

```
from sklearn.metrics import classification_report

print(classification_report(y_test,predictions))

#in this case, we simply use Accuracy of 75% as the metric
#and ignore other metrics
```

```
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         3
           1       0.50      1.00      0.67         1

    accuracy                           0.75         4
   macro avg       0.75      0.83      0.73         4
weighted avg       0.88      0.75      0.77         4
```

E. STEP 5: NEW PREDICTION USING NEW VALUE





III. DIABETES EXAMPLE (2 PREDICTOR VARIABLES)

https://www.alvinang.sg/s/Logistics_Regression_with_Python_Diabetes_Example_by_Dr_Alvin_Ang.ipynb

https://www.alvinang.sg/s/diabetes.csv

References:

https://medium.com/@pragya_paudyal/diabetics-prediction-using-logistic-regression-in-python-e51b90630f2f

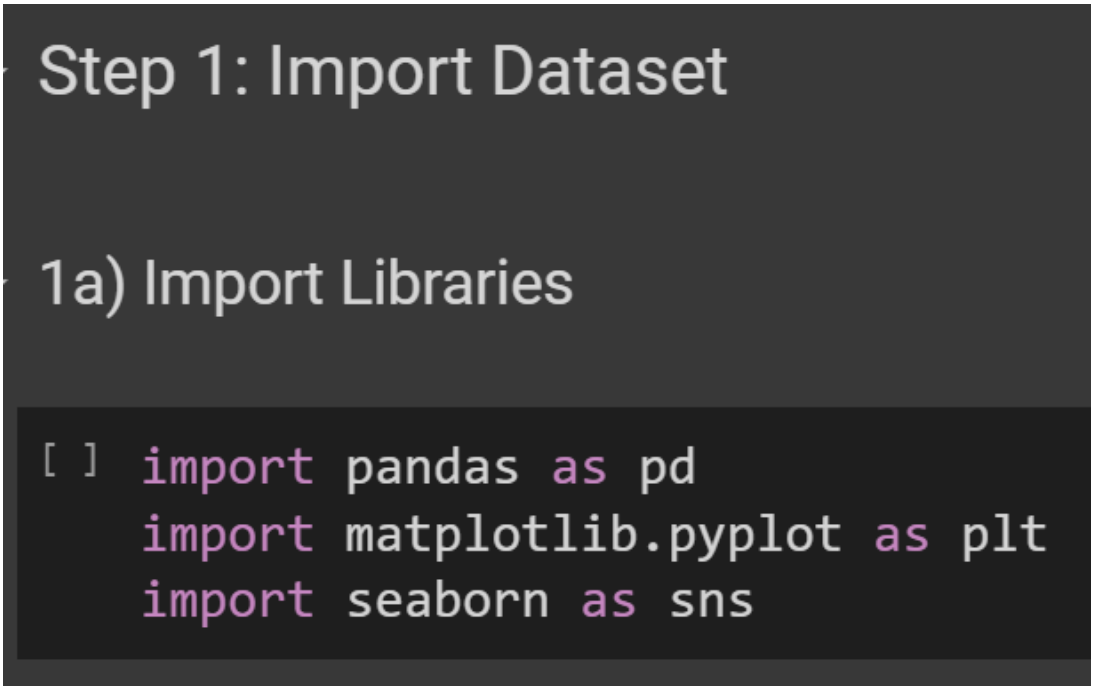
https://www.javatpoint.com/logistic-regression-in-machine-learning

https://medium.com/the-researchers-guide/binary-logistic-regression-using-python-research-oriented-modelling-and-interpretation-49b025f1b510

https://github.com/rahul-raoniar/Rahul_Raoniar_Blogs

A. STEP 1: IMPORT DATASET

1. IMPORT LIBRARIES



2. IMPORT DIABETES DATASET

```
diabetes.sample(5)

#pregnant: Number of times pregnant
#glucose: Plasma glucose concentration (glucose tolerance test)
#pressure: Diastolic blood pressure (mm Hg)
#triceps: Triceps skin fold thickness (mm)
#insulin: 2-Hour serum insulin (mu U/ml)
#mass: Body mass index (weight in kg/(height in m)\²)
#pedigree: Diabetes pedigree function
#age: Age (years)

#diabetes: diabetes case (pos/neg)
```

	pregnant	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
242	2	105	80	45	191	33.7	0.711	29	pos
38	4	123	80	15	176	32.0	0.443	34	neg
82	6	134	70	23	130	35.4	0.542	29	pos
177	12	88	74	40	54	35.3	0.378	48	neg
151	0	137	68	14	148	24.8	0.143	21	neg

3. MAP NEGATIVE TO 0 AND POSITIVE TO 1

1c) Map Negative to 0 and Positive to 1

```
[ ] diabetes['diabetes'] = diabetes['diabetes'].map({'neg':0, 'pos':1})
```

B. STEP 2: PLOTTING

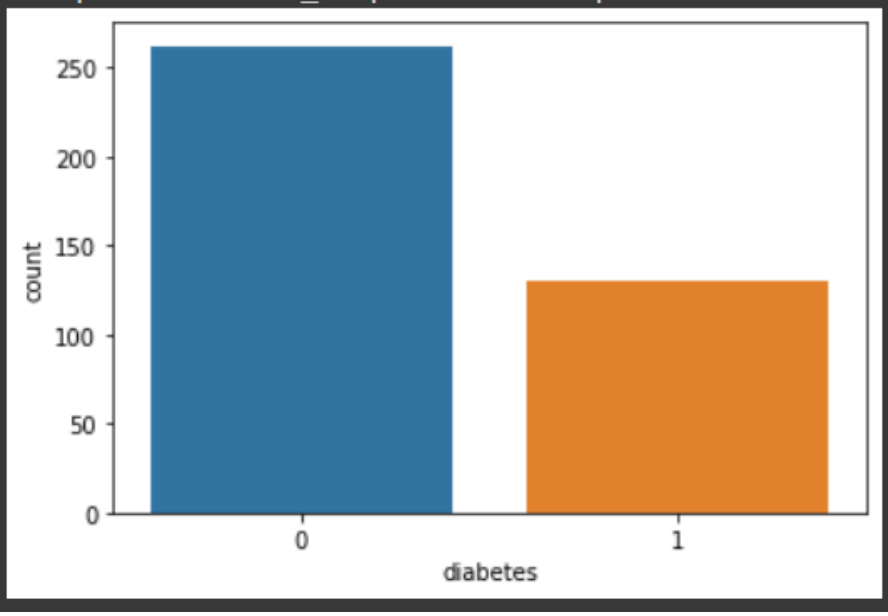
1. COUNTING THE NUMBER OF DIABETICS

Step 2: Plotting

2a) Counting the Number of Diabetics

```
sns.countplot(x='diabetes', data=diabetes)
```

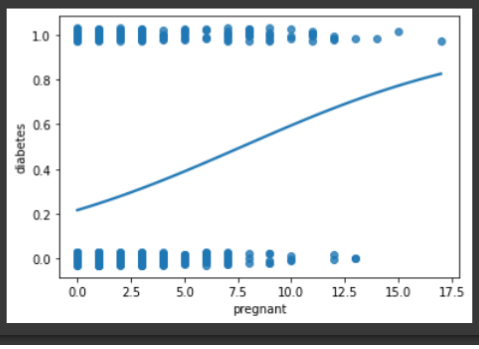
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0bcd0278d0>
```



2. DOES NUMBER OF TIMES BEING PREGNANT CAUSE DIABETES?

2b) Does Number of Times being Pregnant Cause Diabetes?

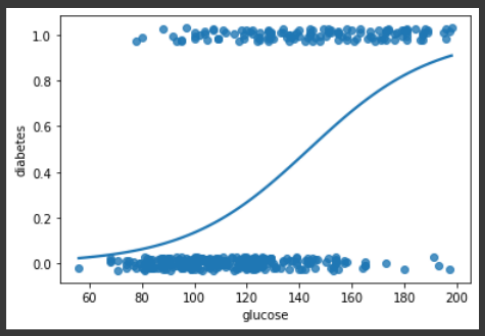
```
sns.regplot(x='pregnant', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

plt.show()
#pregnant: Number of times pregnant
#seems like if you are pregnanre more than 12 times....higher chance
# of being diabetic
```



3. DOES GLUCOSE LEVEL CAUSE DIABETES?

2c) Does Glucose Level cause Diabetes?

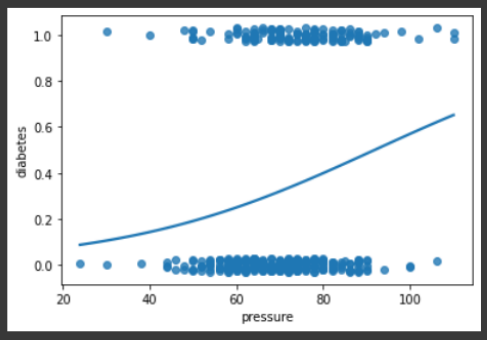
```
sns.regplot(x='glucose', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

plt.show()

#glucose: Plasma glucose concentration (glucose tolerance test)
#the stronger the glucose level, the more likely for diabetes to occur
```



4. DOES BLOOD PRESSURE CAUSE DIABETES?

2d) Does Blood Pressure Cause Diabetes?

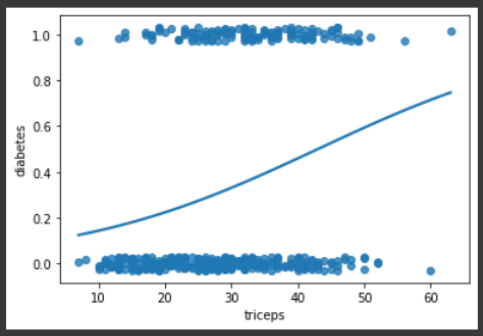
```
[ ] sns.regplot(x='pressure', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

    plt.show()

    #pressure: Diastolic blood pressure (mm Hg)
    #quite hard to gauge...
```



5. DOES TRICEPS SKIN THICKNESS CAUSE DIABETES?

2e) Does Triceps Skin Thickness Cause Diabetes?

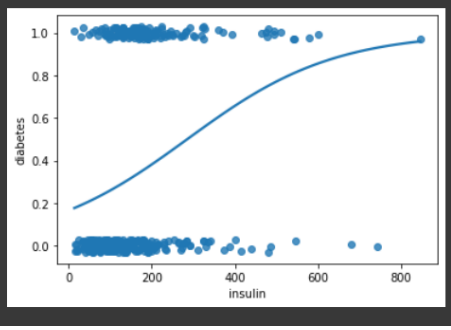
```
[ ] sns.regplot(x='triceps', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

    plt.show()

    #triceps: Triceps skin fold thickness (mm)
    #quite hard to tell...
```



6. DOES INSULIN LEVEL CAUSE DIABETES?

2f) Does Insulin level cause Diabetes?

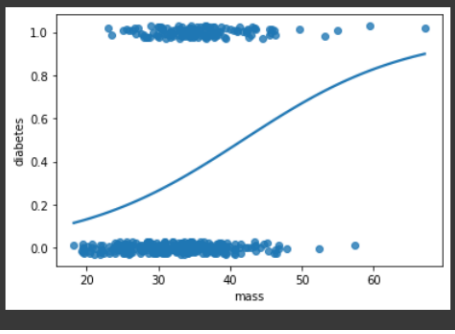
```
sns.regplot(x='insulin', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

plt.show()

#insulin: 2-Hour serum insulin (mu U/ml)
#seems like there maybe some outliers we need to remove (at the ends)
#in order for us to have a better gauge....
```



7. DOES BMI CAUSE DIABETES?

2g) Does BMI Cause Diabetes?

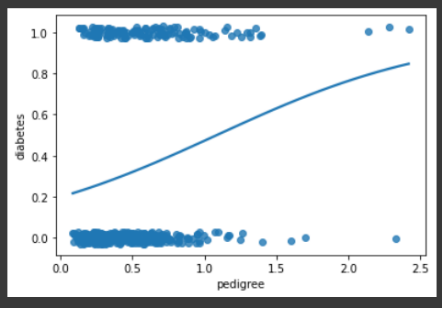
```
sns.regplot(x='mass', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

plt.show()

#mass: Body mass index (weight in kg/(height in m)\²)
#the heavier the mass, the more likelihood for diabetes
```



8. DOES PEDIGREE CAUSE DIABETES?

2h) Does Pedigree Cause Diabetes?

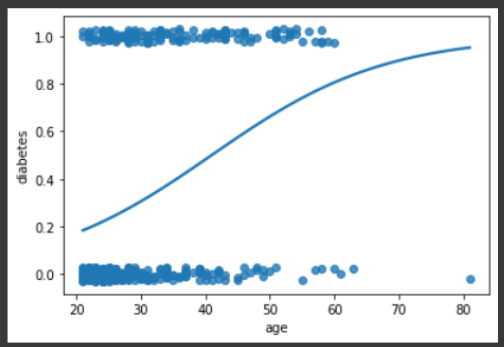
```
sns.regplot(x='pedigree', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

plt.show()

#pedigree: Diabetes pedigree function
#not so sure what Pedigree is...
```



9. DOES AGE CAUSE DIABETES?

2i) Does Age Cause Diabetes?

```
sns.regplot(x='age', y='diabetes', y_jitter=0.03, data=diabetes, logistic = True, ci = None)

plt.show()

#age: Age (years)
#seems like if we removed the outliers (at the ends)
#we might be able to decipher the "Threshold Value" to be around past
#25 years old to have a higher likelihood of diabetes?
```



C. CONCLUSION

Conclusion:

- For now, I've only managed to source out 'Glucose' and 'Mass' that can be predicted using the Logit curve.
- However, I believe that if outliers were removed from most of the other columns / features, the Logistics curve would have fitted better.
- And thereafter we can use Logistics Regression for Predicting those columns.

D. STEP 3: EXTRACT GLUCOSE AND MASS OUT FROM THE DATASET

Step 3: Extract 'Glucose' and 'Mass' out from the Dataset

```
diabetes
```

	pregnant	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
0	1	89	66	23	94	28.1	0.167	21	0
1	0	137	40	35	168	43.1	2.288	33	1
2	3	78	50	32	88	31.0	0.248	26	1
3	2	197	70	45	543	30.5	0.158	53	1
4	1	189	60	23	846	30.1	0.398	59	1
...	...	...	...	...	...	...	...	...	...
387	0	181	88	44	510	43.3	0.222	26	1
388	1	128	88	39	110	36.5	1.057	37	1
389	2	88	58	26	16	28.4	0.766	22	0
390	10	101	76	48	180	32.9	0.171	63	0
391	5	121	72	23	112	26.2	0.245	30	0

392 rows × 9 columns

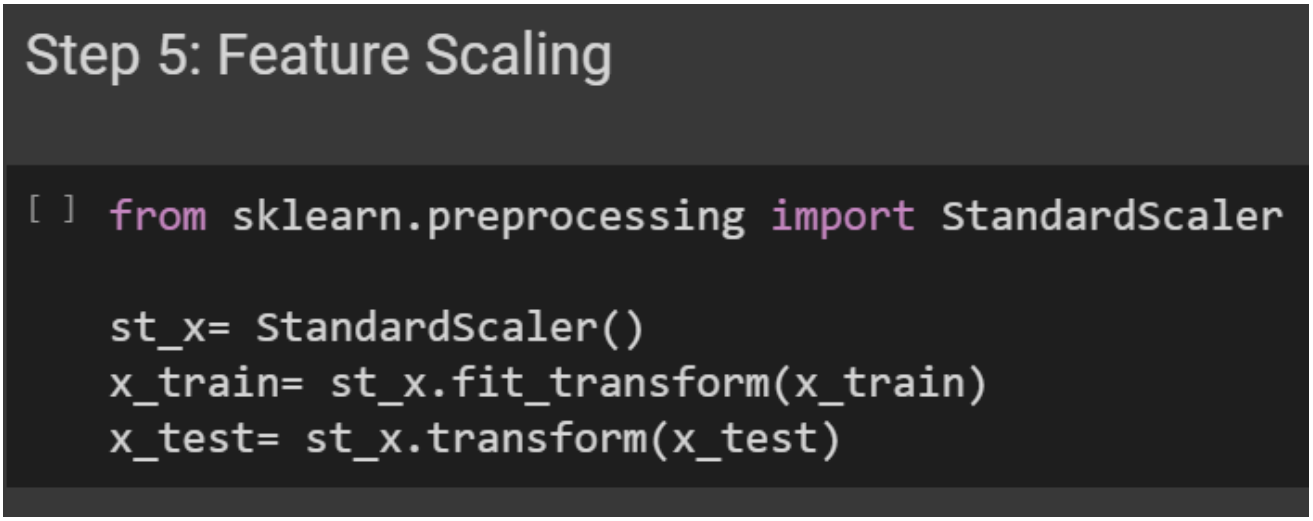
```
x = diabetes.iloc[:, [1,5]].values
y = diabetes.iloc[:,8].values

#we take only Glucose and Mass into accouunt
#for predicting Diabetic or not
```

E. STEP 4: TRAIN TEST SPLIT

Step 4: Train Test Split

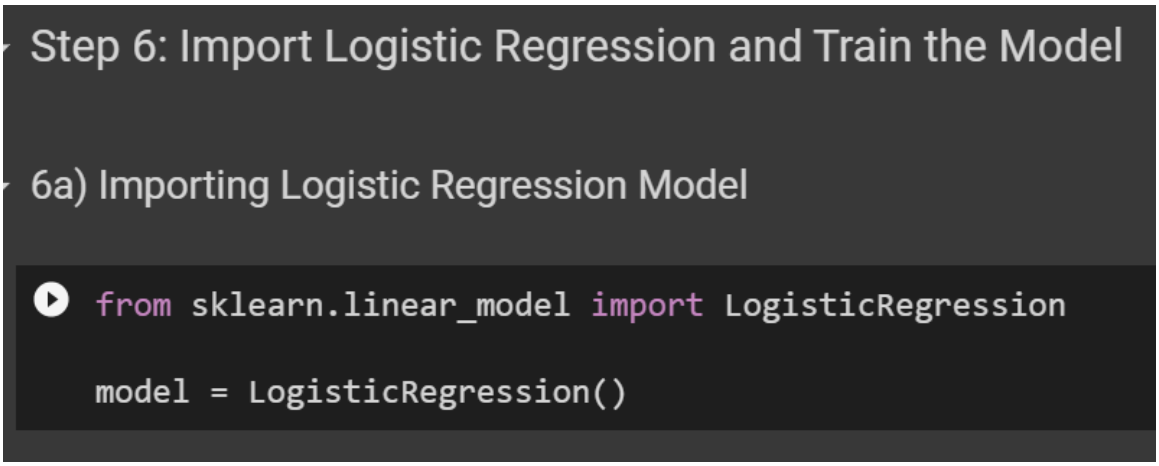
```
[ ] from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)

    #20% test size, 80% training size
```

F. STEP 5: FEATURE SCALING

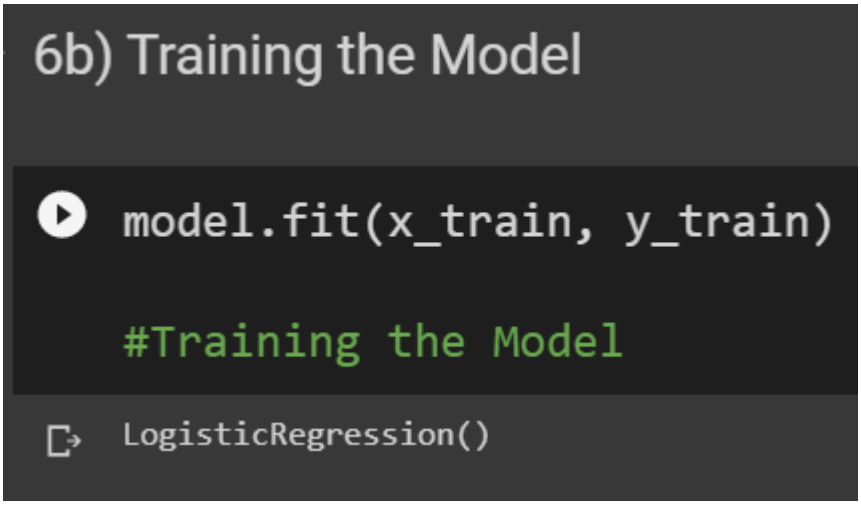


G. STEP 6: IMPORT LOGISTIC REGRESSION AND TRAIN THE MODEL

1. IMPORTING LOGISTIC REGRESSION MODEL



2. TRAINING THE MODEL



3. USING THE MODEL TO PREDICT THE X_TEST

6c) Using the Model to Predict the x_test

```
[ ] #Predicting the test set result
    y_pred= model.predict(x_test)
```

```
[ ] x_test.shape

    #79 rows 2 columns
```

```
(79, 2)
```

```
y_pred
```

```
array([0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])
```

H. STEP 7: METRICS

1. CONFUSION MATRIX

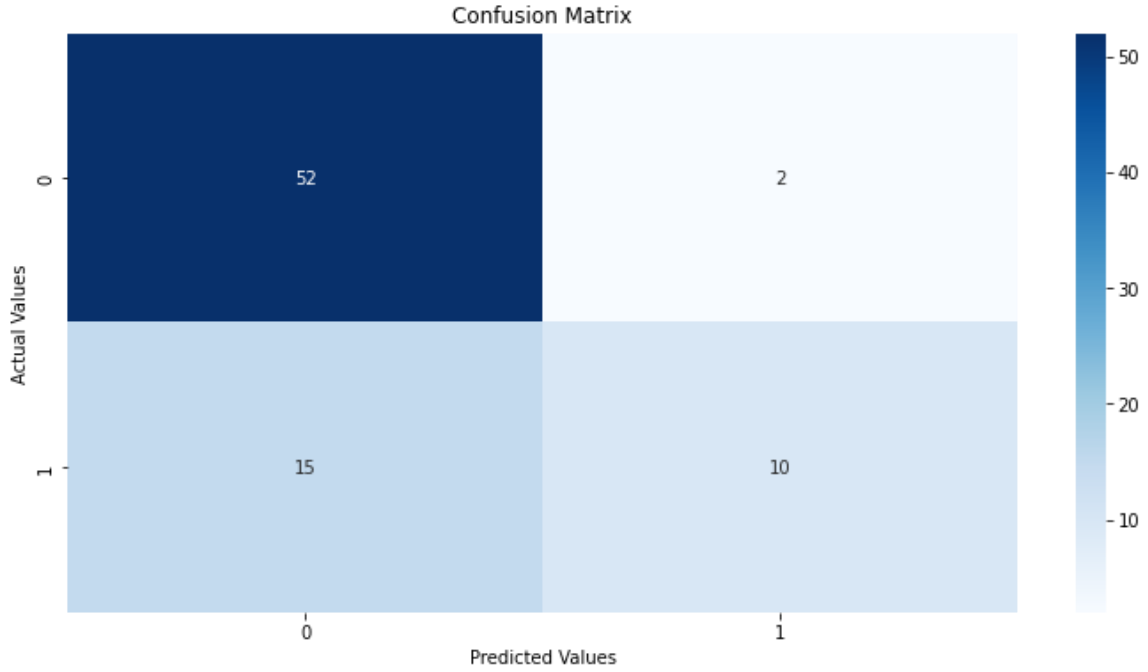
Step 7: Metrics

7a) Confusion Matrix

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

    print(cm)
```

```
[[52  2]
 [15 10]]
```

```
# plotting the confusion matrix
plt.figure(figsize=(12,6))
plt.title("Confusion Matrix")
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.ylabel("Actual Values")
plt.xlabel("Predicted Values")
plt.savefig('confusion_matrix.png')

# 2 Type 1 Errors
# 15 Type 2 Errors
```



2. ACCURACY SCORE

7b) Accuracy Score

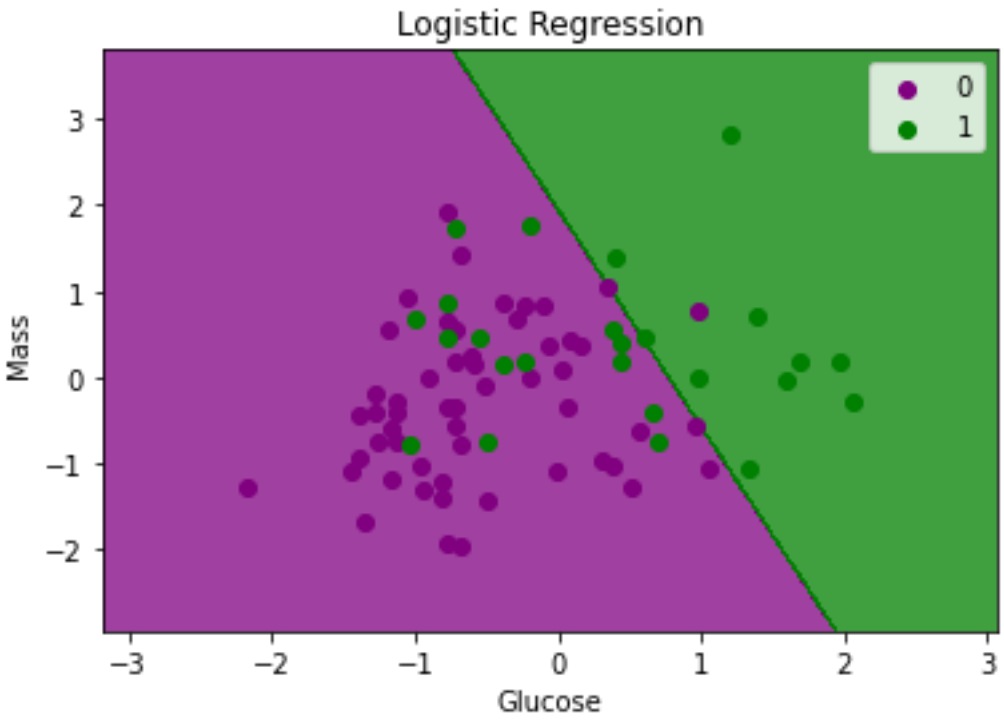
```
[ ] from sklearn.metrics import accuracy_score

    accuracy_score(y_test,y_pred)

    #total = 79 rows of data predicted
    #17 errors / total 79 trials = 0.215
    #1 - 0.215 = 78% accuracy
```

```
0.7848101265822784
```

I. STEP 8: VISUALIZING THE TEST SET RESULT

Step 8: Visualizing the Test Set Result

```
import numpy as np

from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
plt.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)

plt.title('Logistic Regression')
plt.xlabel('Glucose')
plt.ylabel('Mass')
plt.legend()
plt.show()
```



```
[ ] #the picture above is actually a 3D graph but looking from the top
    #(planar view)
    #the Z axis (or 3rd axis) represents 'Diabetic or not' but you can't
    #see it from the chart.

    #we see that Mass doesn't impact Diabetics so much
    #Whether you are High or Low body Mass, it doesn't really determine if you
    #will be diabetic

    #Rather, the core focus is the Glucose level.
    #The higher the Glucose leve, the more certainty for Diabetes.
```

THE END

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.