11/10/2022

Social Network Ads

[Document subtitle]

eman salah eldin
ZEP ANALYTICS

Table of Contents:

1. Introduction: 2
2. About data: 2
3. Analysis and model the data 2
    1. Read data: 2
    2. Prepare and analysis the data: 3
    3. plot the data: 5
    4. split the data: 7
    5. The model: 7
4. Conclusion: 11

1. Introduction:

Machine learning is an exciting field, we use it in different fields of (data science, Ai)

It has three kinds of learning, supervised learning (labeled data): describes a class of problem that involves using a model to learn a mapping between input examples and the target variable, unsupervised learning (unlabeled data): describes a class of problems that involves using a model to describe or extract relationships in data. , reinforcement learning : describes a class of problems where an agent operates in an environment and must *learn* to operate using feedback , predict the next state based on (the current state , the previous action and the reward of this (state-action))

2. About data:

This data is called social network ads, I am downloaded from Kaggle dataset
https://www.kaggle.com/datasets/micheldc55/social-network-ads

Data has information on gender, age and whether the user purchased the product shown to them or not. And I must determine if the user will buy or not

This problem is supervised learning (labeled data), features and class with its values:

1. Gender (Male or Female)
2. Age (18-60)
3. Estimated salary (15000-150000)
4. Purchased (buy or not buy)

3. Analysis and model the data

1. Read data:

I used pandas library from python to read csv file and print the data

```
data.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

2. Prepare and analysis the data

1. normalize the data to make it in the same range: change (Age-Estimated Salary) columns

```
data.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	1	-1.779569	-1.488183	0
1	15810944	1	-0.253270	-1.458854	0
2	15668575	0	-1.111813	-0.784307	0
3	15603246	0	-1.016419	-0.373714	0
4	15804002	1	-1.779569	0.183521	0

2. shuffle the data to make the model trained well

```
[9] data.head()
```

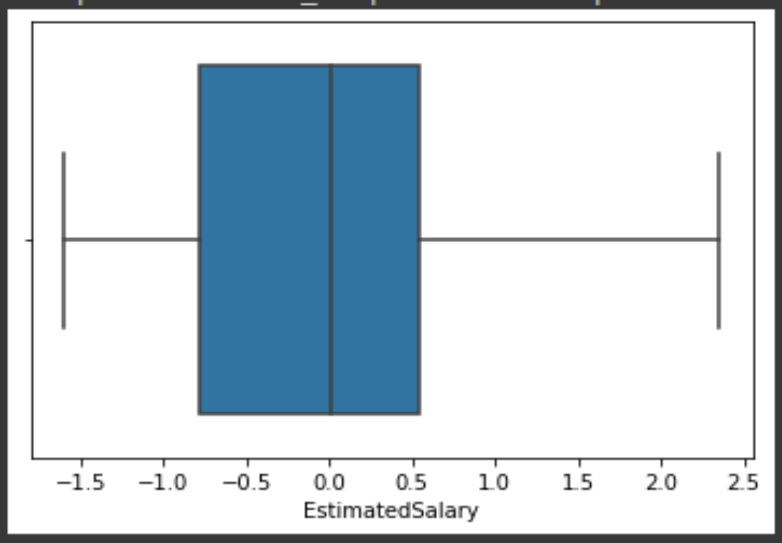
	User ID	Gender	Age	EstimatedSalary	Purchased
60	15814004	1	-1.016419	-1.458854	0
168	15789815	1	-0.825632	2.295146	1
338	15810800	0	0.032911	-0.432370	0
188	15674206	1	-0.253270	0.066208	0
296	15673539	1	0.414485	0.095536	1

3. check the missing values in the data, outliers, duplicated rows in the data
   1. First check missing values:
      there are no missing values in data

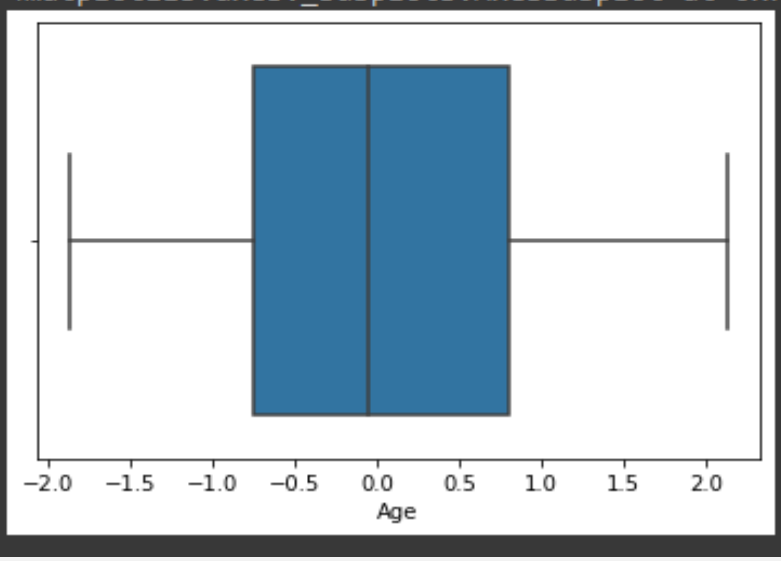
```
[12] data_set.isnull().sum()

Age                0
Gender             0
EstimatedSalary    0
Purchased          0
dtype: int64
```

2. Second check outliers:
   by using boxplot from seaborn library :(Age-Estimated Salary) columns

```
[13] sns.boxplot(data_set['EstimatedSalary'])

/usr/local/lib/python3.7/dist-packages/seaborn/_decor
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a14489
```




```
[14] sns.boxplot(data_set['Age'])

/usr/local/lib/python3.7/dist-packages/seaborn/_decor
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a143bb
```



3. Check duplicated rows:
   There are 20 rows duplicated and solve it by using pandas library with drop duplicates function to remove the 20 rows from the data
   Before:

```
[15] data_set.duplicated().sum()

     20
```

   After:

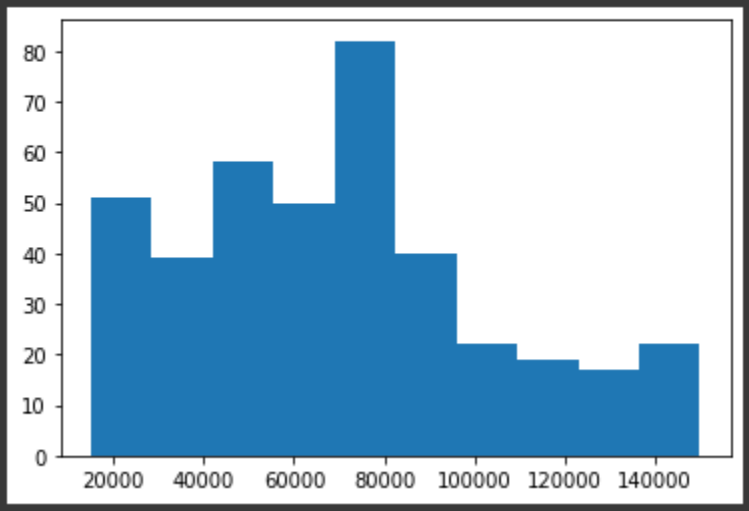
```
[18] data_set.duplicated().sum()

     0
```

3. plot the data:

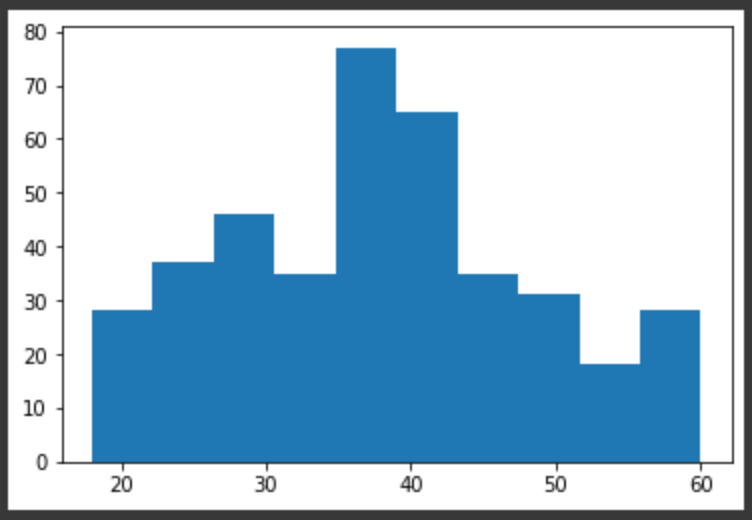
using matplotlib library to plot histogram before and after the normalization for each (Age-Estimated Salary) columns :

Before:

```
[6] plt.hist(data['EstimatedSalary'])
    plt.show()
```

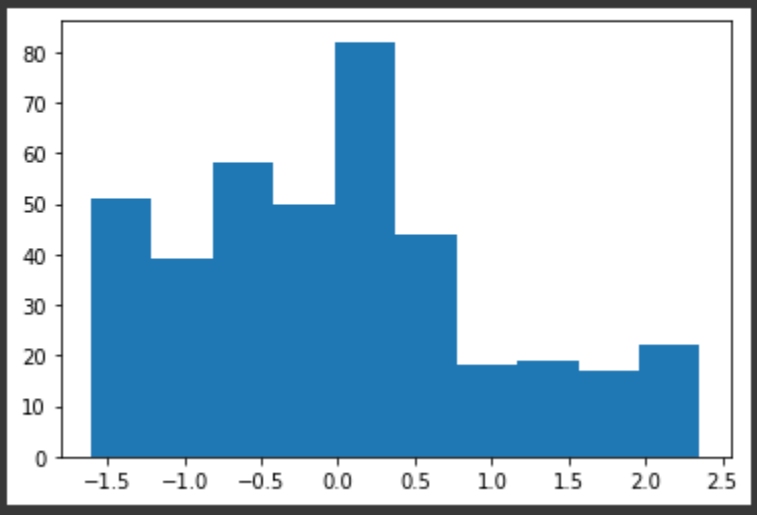


```
plt.hist(data['Age'])
plt.show()
```

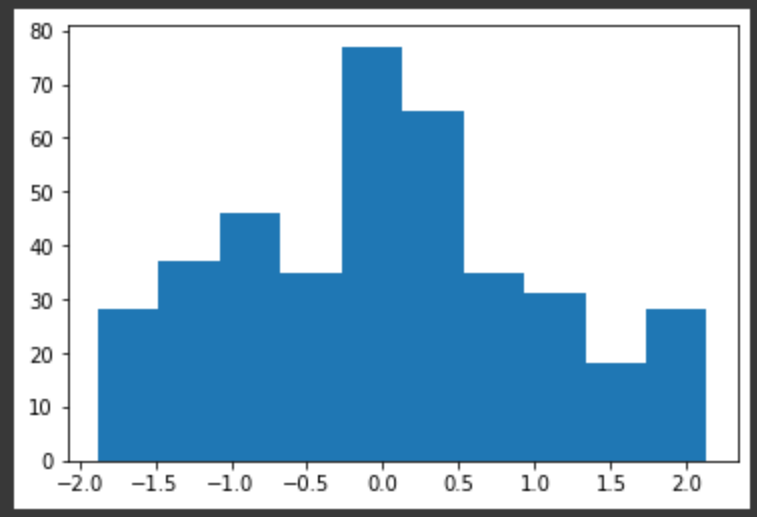


After:

```
plt.hist(data['EstimatedSalary'])
plt.show()
```



```
plt.hist(data['Age'])
plt.show()
```



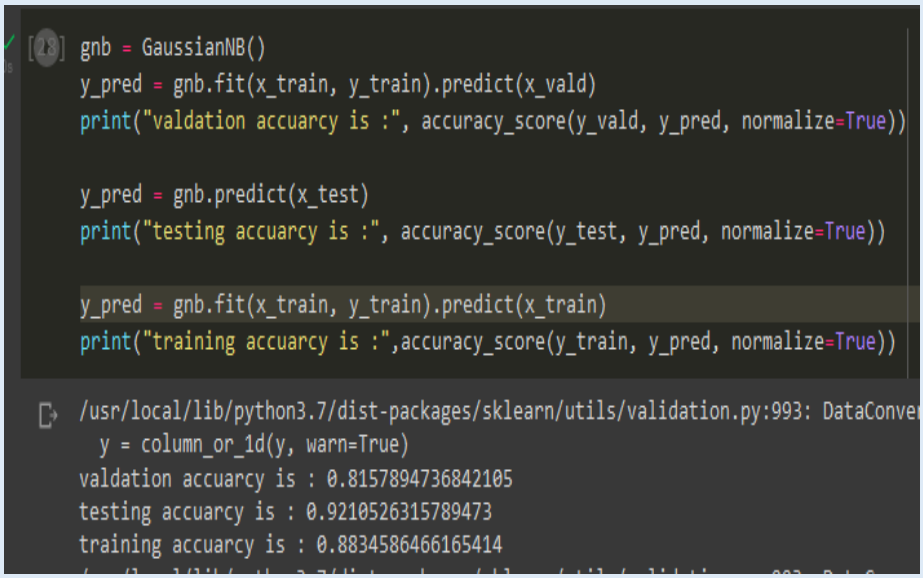
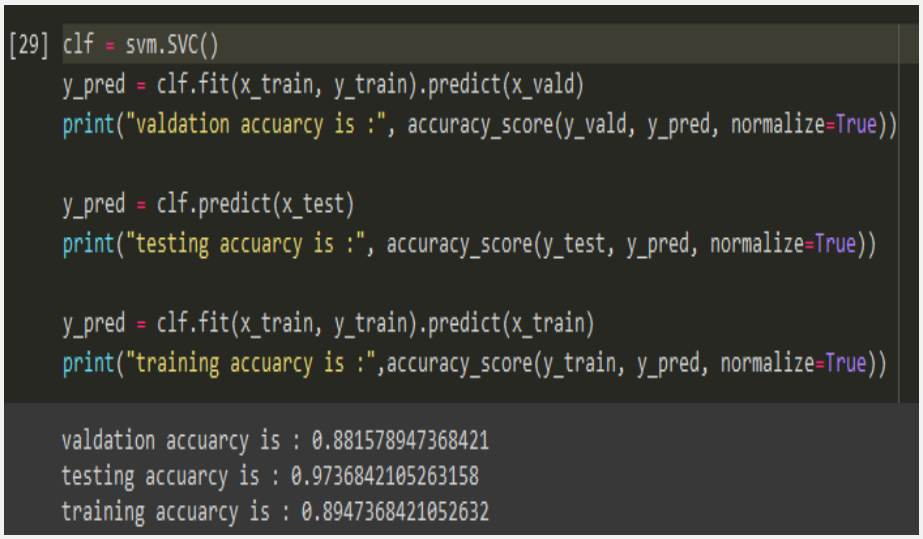
4. split the data:

Usually in machine learning, split data for train validate test, to train the model first and then test the model with unseen data

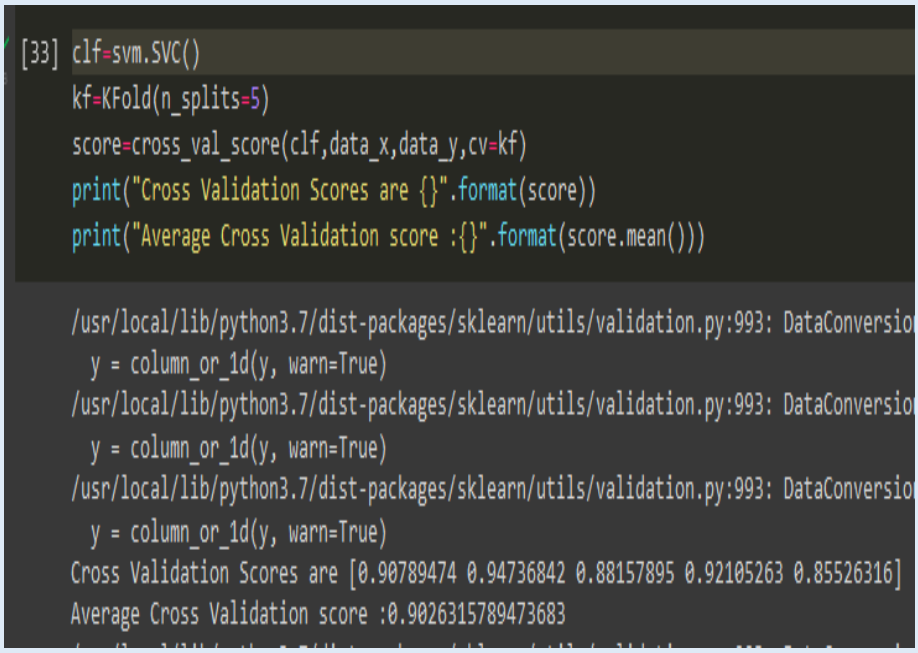
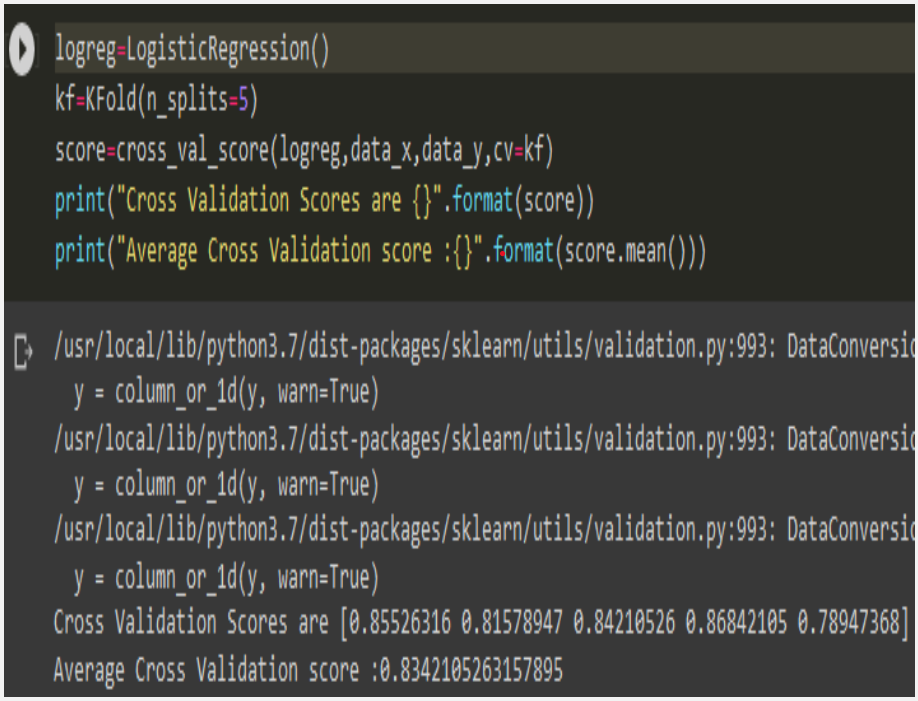
In this model split the data for 70 % training, 20 % validation, 10% test

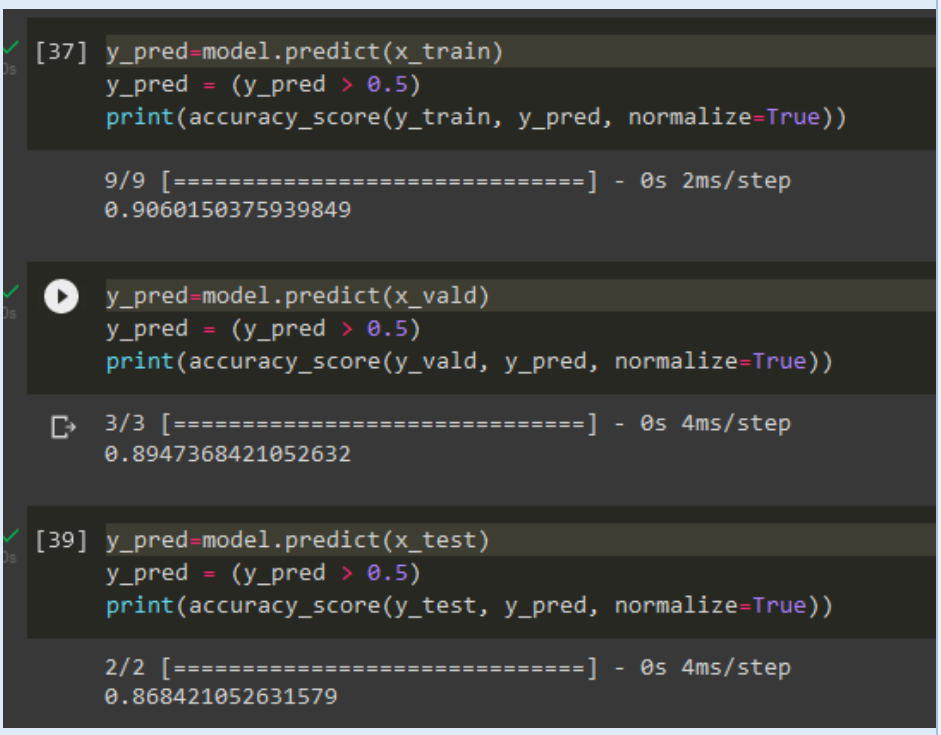
5. The model:

1. Using (Naive bayes, SVM, Logistic regression) classifiers
2. Using cross-validation with k=5 with (Naive bayes, SVM, Logistic regression) I used cross-validation because the data is small, and I try to find a solution by cross-validation
3. Using neural network with 3 layers (input -hidden- output)
    1. Input layer: the number of neurons is the number of the input of the data is 3
    2. Hidden layer: the number of neurons is 16 the best accuracy compared with before it and similar accuracy compared with after it
    3. Output layer: the number of neurons is the number of the classes is 2

Classifier	Accuracy
Naïve Bayes classifier	Training Accuracy: 0.8834586466165414 Validation Accuracy: 0.9078947368421053 Test Accuracy: 0.868421052631579 
SVM	Training Accuracy: 0.8947368421052632 Validation Accuracy: 0.9342105263157895 Test Accuracy: 0.868421052631579 

Logistic Regression	Training Accuracy: 0.8383458646616542 Validation Accuracy: 0.8421052631578947 Test Accuracy: 0.868421052631579 [30] log = LogisticRegression(random_state=42) y_pred = log.fit(x_train, y_train).predict(x_vald) print("valdation accuarcy is :", accuracy_score(y_vald, y_pred, normalize=True)) y_pred = log.predict(x_test) print("testing accuarcy is :", accuracy_score(y_test, y_pred, normalize=True)) y_pred = log.fit(x_train, y_train).predict(x_train) print("training accuarcy is :",accuracy_score(y_train, y_pred, normalize=True)) valdation accuarcy is : 0.8289473684210527 /usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConve y = column_or_1d(y, warn=True) /usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConve y = column_or_1d(y, warn=True) testing accuarcy is : 0.8157894736842105 training accuarcy is : 0.8383458646616542
Naïve bayes with cross-validation in all the data	Average score: 0.881578947368421 gnb=GaussianNB() kf=KFold(n_splits=5) score=cross_val_score(gnb,data_x,data_y,cv=kf) print("Cross Validation Scores are {}".format(score)) print("Average Cross Validation score :{}".format(score.mean())) /usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversion y = column_or_1d(y, warn=True) Cross Validation Scores are [0.90789474 0.89473684 0.88157895 0.88157895 0.84210526] Average Cross Validation score :0.881578947368421

SVM with cross-validation in all the data	`Average score: 0.9026315789473683` 
Logistic Regression with cross-validation in all the data	`Average score: 0.8342105263157895` 

Neural Network	Training Accuracy: 0.9060150375939849 Validation Accuracy: 0.8947368421052632 Test Accuracy: 0.868421052631579 

```
[37] y_pred=model.predict(x_train)
     y_pred = (y_pred > 0.5)
     print(accuracy_score(y_train, y_pred, normalize=True))

     9/9 [==============================] - 0s 2ms/step
     0.9060150375939849

     y_pred=model.predict(x_vald)
     y_pred = (y_pred > 0.5)
     print(accuracy_score(y_vald, y_pred, normalize=True))

     3/3 [==============================] - 0s 4ms/step
     0.8947368421052632

[39] y_pred=model.predict(x_test)
     y_pred = (y_pred > 0.5)
     print(accuracy_score(y_test, y_pred, normalize=True))

     2/2 [==============================] - 0s 4ms/step
     0.868421052631579
```

4. Conclusion:

After building these models and looking at their accuracies, the best model is (SVM and neural network) trained well and predict high accuracy in train validation and test accuracy, and the model with the least accuracy is logistic regression