CYCLE - 4

1. Using the iris data set, implement the KNN algorithm. Take different values for the Test and training data set .Also use different values for k. Also find the accuracy level.

CODE:

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read_csv("iris.csv")

x = dataset.iloc[:,:-1].values

y = dataset.iloc[:,4].values

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=5)

classifier.fit(x_train,y_train)

y_pred=classifier.predict(x_test)

from sklearn.metrics import classification_report,confusion_matrix

print(classification_report(y_test,y_pred))
```

OUTPUT:

K=5, TEST= 0.20, TRAIN= 0.80

	precision	recall	f1-score	support
Setosa	1.00	1.00	1.00	16
Versicolor	1.00	1.00	1.00	8
Virginica	1.00	1.00	1.00	6
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

K=3, TEST= 0.20, TRAIN= 0.80

	precision	recall	f1-score	support
Setosa	1.00	1.00	1.00	9
Versicolor	0.93	1.00	0.96	13
Virginica	1.00	0.88	0.93	8
accuracy			0.97	30
macro avg	0.98	0.96	0.97	30
weighted avg	0.97	0.97	0.97	30

2. Download another data set suitable for the KNN and implement the KNN algorithm. Take different values for the Test and training data set .Also use different values for k.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv("Blood Transfusion.csv")
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,4].values
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_pred))
```

OUTPUT:

```
              precision    recall  f1-score   support

           0       0.83      0.93      0.87       120
           1       0.44      0.23      0.30        30

    accuracy                           0.79       150
   macro avg       0.63      0.58      0.59       150
weighted avg       0.75      0.79      0.76       150
```

3. Using iris data set, implement naive bayes classification for different naive Bayes classification algorithms.((i) gaussian (ii) bernoulli etc)

- Find out the accuracy level w.r.t to each algorithm
- Display the no:of mislabeled classification from test data set
- List out the class labels of the mismatching records

I. Gaussian

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('iris.csv')
x=dataset.iloc[:,:4].values
y=dataset['variety'].values
dataset.head(5)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
print(y_pred)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test,y_pred))
df=pd.DataFrame({'Real values':y_test,'Predicted values':y_pred})
print(df)
```

OUTPUT:

```
['Setosa' 'Setosa' 'Virginica' 'Versicolor' 'Setosa' 'Virginica'
 'Virginica' 'Virginica' 'Versicolor' 'Setosa' 'Virginica' 'Versicolor'
 'Setosa' 'Versicolor' 'Setosa' 'Virginica' 'Setosa' 'Versicolor' 'Setosa'
 'Versicolor' 'Setosa' 'Versicolor' 'Versicolor' 'Setosa' 'Virginica'
 'Versicolor' 'Versicolor' 'Versicolor' 'Setosa' 'Virginica' 'Versicolor'
 'Virginica' 'Setosa' 'Setosa' 'Setosa' 'Virginica' 'Setosa' 'Setosa'
 'Virginica' 'Setosa' 'Versicolor' 'Virginica' 'Setosa' 'Setosa'
 'Virginica']
[[19  0  0]
 [ 0 13  1]
 [ 0  0 12]]
Accuracy :  0.9777777777777777
   Real values Predicted values
0       Setosa           Setosa
1       Setosa           Setosa
2    Virginica        Virginica
3   Versicolor       Versicolor
4       Setosa           Setosa
5    Virginica        Virginica
6    Virginica        Virginica
7    Virginica        Virginica
8   Versicolor       Versicolor
9       Setosa           Setosa
10   Virginica        Virginica
11  Versicolor       Versicolor
12      Setosa           Setosa
13  Versicolor       Versicolor
14      Setosa           Setosa
15   Virginica        Virginica
16      Setosa           Setosa
```

```
16      Setosa           Setosa
17  Versicolor       Versicolor
18      Setosa           Setosa
19  Versicolor       Versicolor
20      Setosa           Setosa
21  Versicolor       Versicolor
22  Versicolor       Versicolor
23      Setosa           Setosa
24   Virginica        Virginica
25  Versicolor       Versicolor
26  Versicolor       Versicolor
27  Versicolor       Versicolor
28      Setosa           Setosa
29   Virginica        Virginica
30  Versicolor       Versicolor
31   Virginica        Virginica
32      Setosa           Setosa
33      Setosa           Setosa
34      Setosa           Setosa
35   Virginica        Virginica
36      Setosa           Setosa
37      Setosa           Setosa
38   Virginica        Virginica
39      Setosa           Setosa
40  Versicolor       Versicolor
41  Versicolor        Virginica
42      Setosa           Setosa
43      Setosa           Setosa
44   Virginica        Virginica
```

II. Bernoulli

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset=pd.read_csv('iris.csv')
x=dataset.iloc[:,:4].values
y=dataset['variety'].values
dataset.head(5)
```

```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
from sklearn.naive_bayes import BernoulliNB
classifier=BernoulliNB()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
print(y_pred)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test,y_pred))
df=pd.DataFrame({'Real values':y_test,'Predicted values':y_pred})
print(df)
```

OUTPUT:

```
['Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor' 'Versicolor']
[[ 0 15  0]
 [ 0 13  0]
 [ 0 17  0]]
Accuracy :  0.28888888888888886
   Real values Predicted values
0       Setosa       Versicolor
1    Virginica       Versicolor
2    Virginica       Versicolor
3       Setosa       Versicolor
4   Versicolor       Versicolor
5   Versicolor       Versicolor
6    Virginica       Versicolor
7    Virginica       Versicolor
8       Setosa       Versicolor
9    Virginica       Versicolor
10  Versicolor       Versicolor
11      Setosa       Versicolor
12      Setosa       Versicolor
13      Setosa       Versicolor
14   Virginica       Versicolor
15  Versicolor       Versicolor
15  Versicolor       Versicolor
16      Setosa       Versicolor
17  Versicolor       Versicolor
18  Versicolor       Versicolor
19   Virginica       Versicolor
20  Versicolor       Versicolor
21  Versicolor       Versicolor
22  Versicolor       Versicolor
23      Setosa       Versicolor
24   Virginica       Versicolor
25  Versicolor       Versicolor
26   Virginica       Versicolor
27   Virginica       Versicolor
28   Virginica       Versicolor
29   Virginica       Versicolor
30  Versicolor       Versicolor
31   Virginica       Versicolor
32      Setosa       Versicolor
33   Virginica       Versicolor
34      Setosa       Versicolor
35      Setosa       Versicolor
36      Setosa       Versicolor
37      Setosa       Versicolor
38   Virginica       Versicolor
39   Virginica       Versicolor
40  Versicolor       Versicolor
41      Setosa       Versicolor
42   Virginica       Versicolor
43      Setosa       Versicolor
44  Versicolor       Versicolor
```

4. Use car details CSV file and implement decision tree algorithm

   - Find out the accuracy level.
   - Display the no: of mislabelled classification from test data set
   - List out the class labels of the mismatching records

CODE:

```
import pandas as pd
data = pd.read_csv('car.csv')
print(data.head())
col_names = ['buying','maint','doors','persons','lug_boot','safety','class']
data.columns = col_names
print(col_names)
#The decision trees implemented in scikit-learn uses only numerical
# features and these features are interpreted
# always as continuous numeric variables.
# pandas.factorize() method helps to get the numeric representation
# of an array by identifying distinct values

data['class'],class_names = pd.factorize(data['class'])
data['buying'],_ = pd.factorize(data['buying'])
data['maint'],_ = pd.factorize(data['maint'])
data['doors'],_ = pd.factorize(data['doors'])
data['persons'],_ = pd.factorize(data['persons'])
data['lug_boot'],_ = pd.factorize(data['lug_boot'])
data['safety'],_ = pd.factorize(data['safety'])

print(data.head())
x = data.iloc[:, :-1]
#print(x)
y = data.iloc[:, -1]
#print(y)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
from sklearn.tree import DecisionTreeClassifier
tree1 = DecisionTreeClassifier()
tree1.fit(x_train,y_train)
y_pred = tree1.predict(x_test)
#how did our model perform?
```

```
count_missclassified = (y_test != y_pred).sum()
print('Misclassified samples count : ',count_missclassified)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
print("Accuracy",accuracy)
```

OUTPUT:

```
  vhigh vhigh.1  2 2.1  small   low  unacc
0  vhigh   vhigh  2   2  small   med  unacc
1  vhigh   vhigh  2   2  small  high  unacc
2  vhigh   vhigh  2   2    med   low  unacc
3  vhigh   vhigh  2   2    med   med  unacc
4  vhigh   vhigh  2   2    med  high  unacc
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
   buying  maint  doors  persons  lug_boot  safety  class
0       0      0      0        0         0       0      0
1       0      0      0        0         0       1      0
2       0      0      0        0         1       2      0
3       0      0      0        0         1       0      0
4       0      0      0        0         1       1      0
Misclassified samples count :  16
Accuracy 0.9691714836223507
```

5. Implement Simple and multiple linear regression for the data sets ‘student_score.csv’ and ‘company_data .csv’ respectively

CODE (Single linear Regression) :

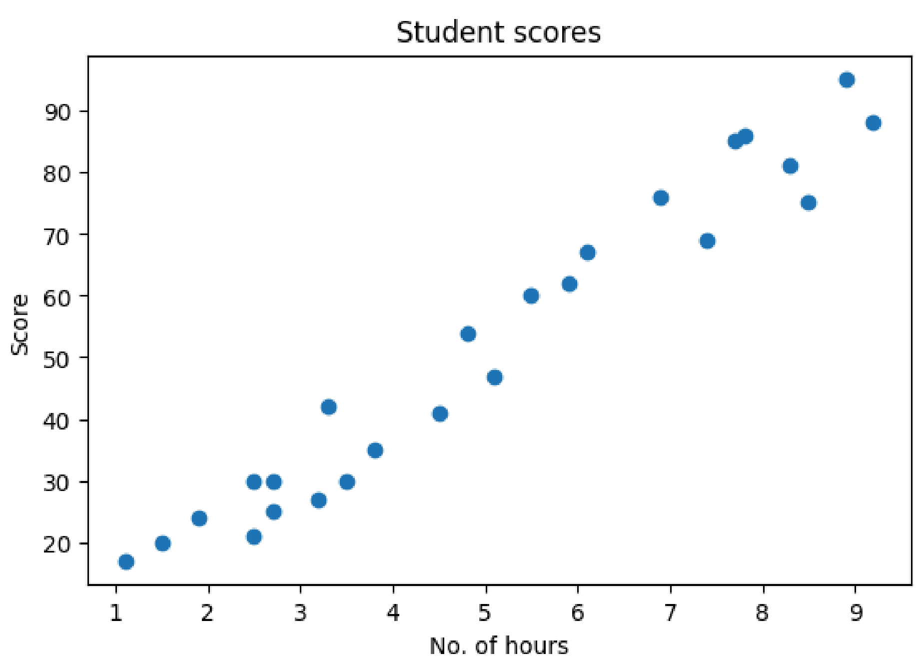
```
import numpy as np
import pandas as pd
student = pd.read_csv('student_scores.csv')
print(student.head())
student.describe()
student.info()
import matplotlib.pyplot as plt

Xax = student.iloc[:,0]
Yax = student.iloc[:,1]
plt.scatter(Xax,Yax)
plt.xlabel("No. of hours")
plt.ylabel("Score")
plt.title("Student scores")
plt.show()
x = student.iloc[:,:-1]
y = student.iloc[:,1]
print('x values : ')
print(x)
print('y values : ')
print(y)
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)
print(x_train)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
print('INTERCEPT = ',regressor.intercept_)
print('CO-EFFICIENT = ',regressor.coef_)
y_pred = regressor.predict(x_test)
for(i,j) in zip(y_test,y_pred):
   if(i!=j):
       print("Actual value : ",i,"Predicted value : ",j)
print("Number of mislabeled points from test data set : ", (y_test !=
y_pred).sum())
```

OUTPUT :

```
   Hours  Scores
0    2.5      21
1    5.1      47
2    3.2      27
3    8.5      75
4    3.5      30
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
x values :
    Hours
0     2.5
1     5.1
2     3.2
3     8.5
4     3.5
5     1.5
6     9.2
7     5.5
8     8.3
9     2.7
10    7.7
11    5.9
12    4.5
```

```
12    4.5
13    3.3
14    1.1
15    8.9
16    2.5
17    1.9
18    6.1
19    7.4
20    2.7
21    4.8
22    3.8
23    6.9
24    7.8
y values :
0     21
1     47
2     27
3     75
4     30
5     20
6     88
7     60
8     81
9     25
10    85
11    62
12    41
13    42
14    17
15    95
16    30
```

```
19    7.4
6     9.2
10    7.7
11    5.9
2     3.2
20    2.7
22    3.8
1     5.1
4     3.5
14    1.1
8     8.3
23    6.9
24    7.8
5     1.5
18    6.1
16    2.5
15    8.9
21    4.8
17    1.9
7     5.5
INTERCEPT =  2.612204693008323
CO-EFFICIENT =  [9.93021104]
Actual value :  75 Predicted value :  87.01899854838967
Actual value :  25 Predicted value :  29.423774505894166
Actual value :  42 Predicted value :  35.381901130979905
Actual value :  41 Predicted value :  47.29815438115139
Actual value :  21 Predicted value :  27.43773229753225
Number of mislabeled points from test data set :  5
Mean Absolute error : 7.159351720397515
Mean Squared error : 57.78729707376474
Root Mean Squared error : 7.6017956479877
```



CODE (Multiple linear regression) :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
advertising = pd.read_csv('Company_data.csv')
advertising.head()
advertising.describe()
advertising.info()
print("Feature values : ")
x = advertising.iloc[:, :-1]
print(x)
print("Target variable values : ")
y = advertising.iloc[:, -1]
print(y)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
print("intercept is : ")
print(regressor.intercept_)
print("Co-efficients are : ")
print(regressor.coef_)
y_pred = regressor.predict(x_test)
for(i,j) in zip(y_test,y_pred):
   if i!=j:
       print("Actual values : ",i," Predicted values : ",j)
print("Number of mislabeled points from test data set : ",(y_test !=
y_pred).sum())
#MSE,MAE,RMSE and R-Squared are mainly used metrics to evaluate
#the prediction error rates and model performance in regression
analysis.
from sklearn import metrics
print("Mean Absolute error :",
metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squared error :", metrics.mean_squared_error(y_test,y_pred))
```

```
print("Root Mean Squared error :",
np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

OUTPUT :

```
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
Feature values :
        TV  Radio  Newspaper
0    230.1   37.8       69.2
1     44.5   39.3       45.1
2     17.2   45.9       69.3
3    151.5   41.3       58.5
4    180.8   10.8       58.4
..     ...    ...        ...
195   38.2    3.7       13.8
196   94.2    4.9        8.1
197  177.0    9.3        6.4
198  283.6   42.0       66.2
199  232.1    8.6        8.7
```

```
[200 rows x 3 columns]
Target variable values :
0      22.1
1      10.4
2      12.0
3      16.5
4      17.9
       ...
195     7.6
196    14.0
197    14.8
198    25.5
199    18.4
Name: Sales, Length: 200, dtype: float64
intercept is :
4.528450435723583
Co-efficients are :
[ 0.05466747  0.1117412  -0.0002813 ]
```

```
Actual values :  14.0  Predicted values :  13.447976708088397
Actual values :  16.0  Predicted values :  18.318509452905268
Actual values :  17.0  Predicted values :  18.939637902572528
Actual values :  17.2  Predicted values :  16.409281011965177
Actual values :  16.0  Predicted values :  15.0990544646798
Actual values :  22.3  Predicted values :  21.278146220965652
Actual values :  6.9  Predicted values :  6.204768961696529
Actual values :  1.6  Predicted values :  8.989222053373664
Actual values :  17.4  Predicted values :  18.94622073271653
Actual values :  6.6  Predicted values :  9.287355769826737
Actual values :  10.6  Predicted values :  10.60671466067842
Actual values :  12.0  Predicted values :  11.767960988436206
Actual values :  10.9  Predicted values :  11.105637500633627
Actual values :  13.2  Predicted values :  10.00130165618592
Actual values :  11.0  Predicted values :  8.998690627131847
Actual values :  20.8  Predicted values :  22.703465125752615
Actual values :  8.7  Predicted values :  8.229332686829284
Actual values :  19.9  Predicted values :  16.843116971804193
Actual values :  21.5  Predicted values :  21.196276213226028
Actual values :  20.2  Predicted values :  21.476724859954288
Actual values :  25.5  Predicted values :  24.70665444012566
Actual values :  16.6  Predicted values :  18.08155360471116
Actual values :  23.2  Predicted values :  22.275233287701035
Actual values :  21.8  Predicted values :  21.98121841035531
Actual values :  11.8  Predicted values :  11.682262577123913
Actual values :  17.6  Predicted values :  14.693845221839664
Actual values :  15.9  Predicted values :  15.50112329945018
Actual values :  5.6  Predicted values :  7.01279372196412
Actual values :  22.6  Predicted values :  21.00696045154617
Actual values :  9.7  Predicted values :  9.115402255705765
Actual values :  13.2  Predicted values :  13.286345287949693
```

```
Actual values :  14.6  Predicted values :  15.24621124737743
Actual values :  10.3  Predicted values :  12.408073568263916
Actual values :  10.9  Predicted values :  10.43701469882918
Actual values :  13.4  Predicted values :  13.761007711902913
Actual values :  26.2  Predicted values :  25.035490254857223
Actual values :  10.4  Predicted values :  11.339895688395877
Actual values :  25.4  Predicted values :  25.07786790253533
Actual values :  18.0  Predicted values :  17.576577079105668
Actual values :  16.7  Predicted values :  15.857709761004456
Actual values :  17.6  Predicted values :  20.774703878787804
Actual values :  14.0  Predicted values :  12.415207950084081
Actual values :  23.8  Predicted values :  24.7597120988801
Actual values :  12.9  Predicted values :  13.77446187102004
Actual values :  12.4  Predicted values :  12.119921031778599
Actual values :  16.7  Predicted values :  14.524214898408742
Actual values :  19.6  Predicted values :  18.339102949129643
Actual values :  20.7  Predicted values :  21.381475079685885
Actual values :  22.1  Predicted values :  21.311787638919434
Actual values :  19.8  Predicted values :  20.96884287777423
Actual values :  13.6  Predicted values :  13.230174929788635
Actual values :  25.4  Predicted values :  24.01205740712085
Actual values :  13.3  Predicted values :  13.199919922631574
Actual values :  14.8  Predicted values :  15.2419861468864
Actual values :  21.4  Predicted values :  23.69067735979671
Actual values :  12.0  Predicted values :  10.578158129553994
Actual values :  19.6  Predicted values :  20.27500482966528
Actual values :  11.3  Predicted values :  9.339446148742997
Number of mislabeled points from test data set :  60
Mean Absolute error : 1.2607485335180788
Mean Squared error : 2.932494017740453
Root Mean Squared error : 1.712452632261825
```

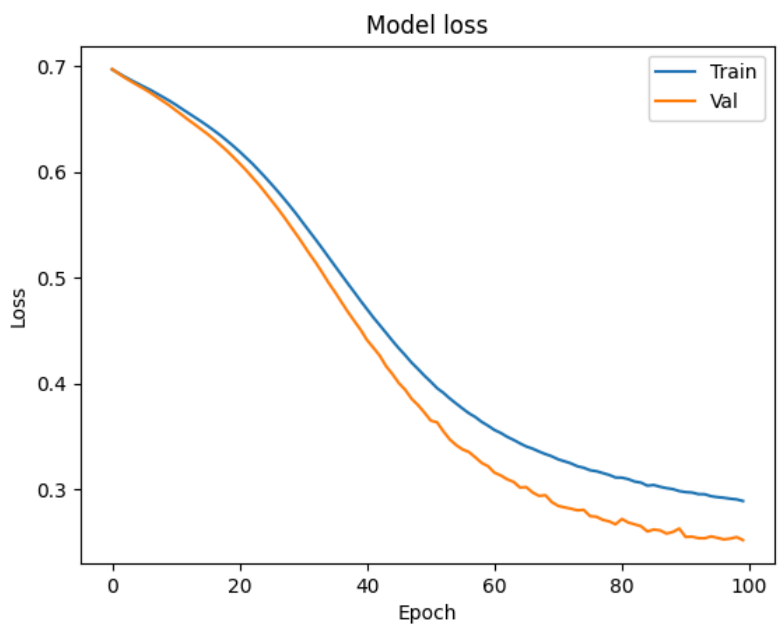
6. Create a neural network for the given ‘houseprice.csv’ to predict the weather price of the house is above or below median value or not

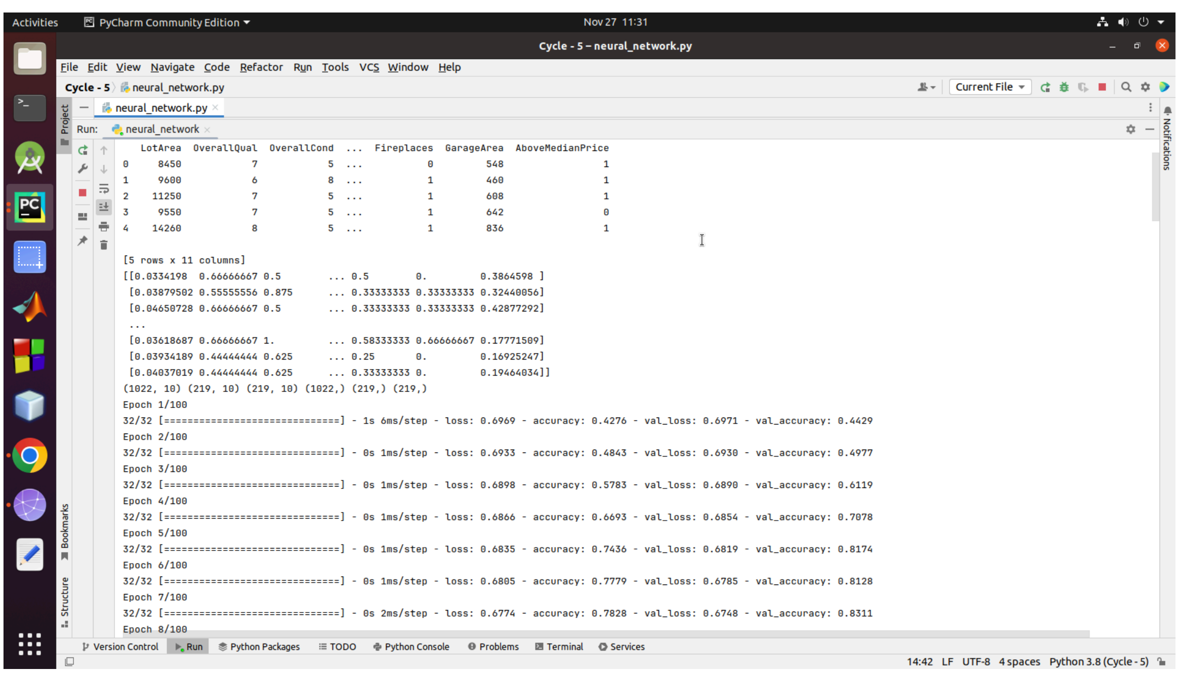
CODE:

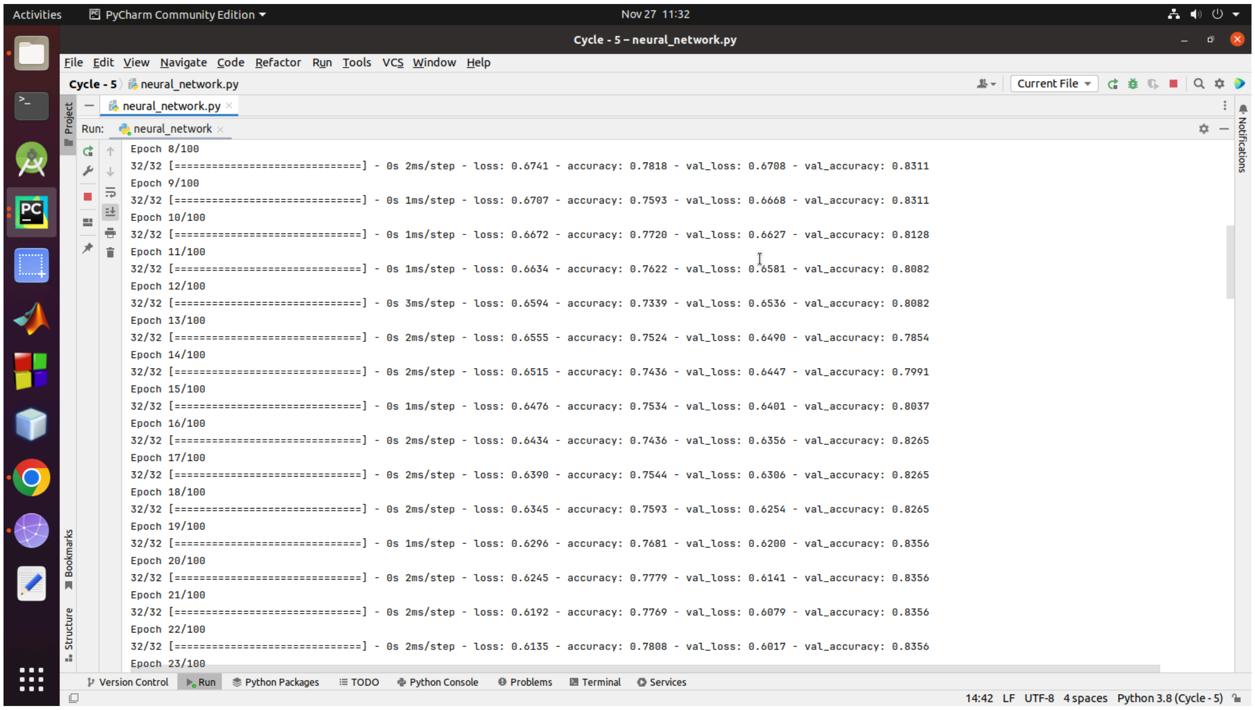
```
import tensorflow as tf
import keras
import pandas
import sklearn
import matplotlib
import pandas as pd
df = pd.read_csv('housepricedata.csv')
print(df.head())
dataset = df.values
X = dataset[:,0:10]
Y = dataset[:,10]
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
print(X_scale)
from sklearn.model_selection import train_test_split
X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale,
Y, test_size=0.3)
X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test,
Y_val_and_test, test_size=0.5)
print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape,
Y_test.shape)
from keras.models import Sequential
from keras.layers import Dense
model = Sequential([Dense(32, activation='relu', input_shape=(10,)), Dense(32,
activation='relu'),Dense(1, activation='sigmoid'),])
model.compile(optimizer='sgd', loss='binary_crossentropy',
metrics=['accuracy'])
hist = model.fit(X_train, Y_train, batch_size=32, epochs=100,
validation_data=(X_val, Y_val))
model.evaluate(X_test, Y_test)[1]

import matplotlib.pyplot as plt
plt.plot(hist.history['loss'])
```

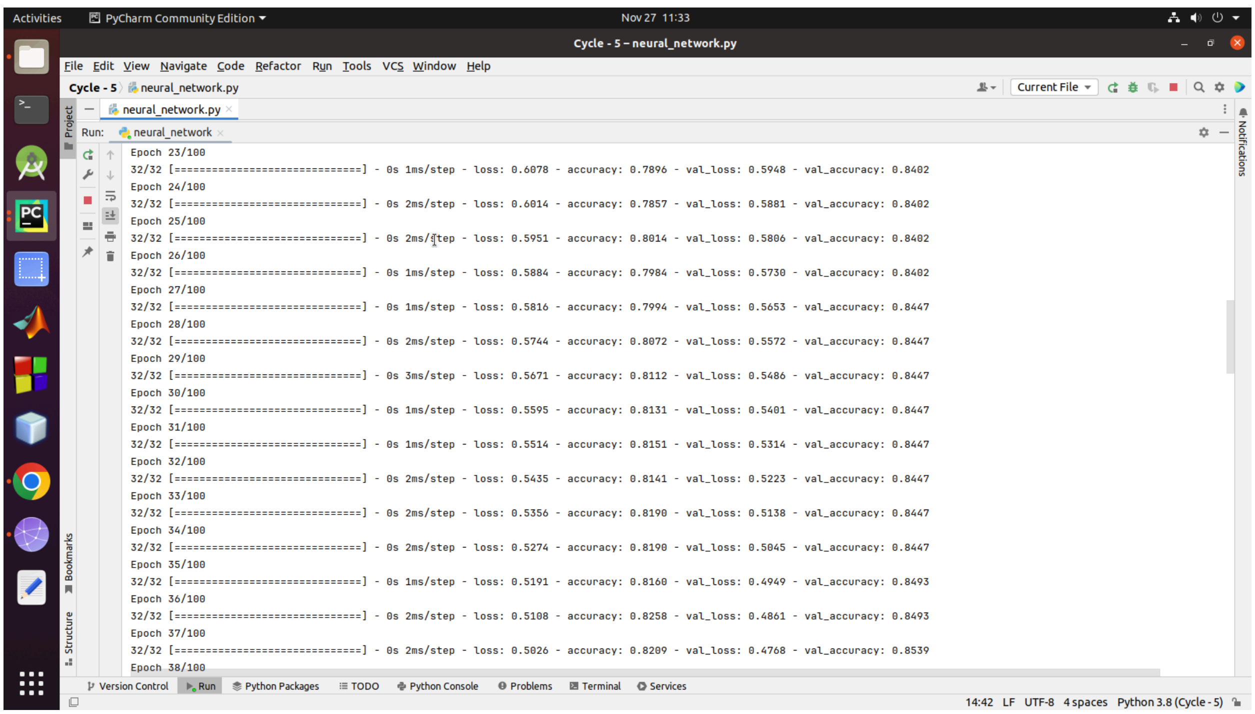
```python
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```

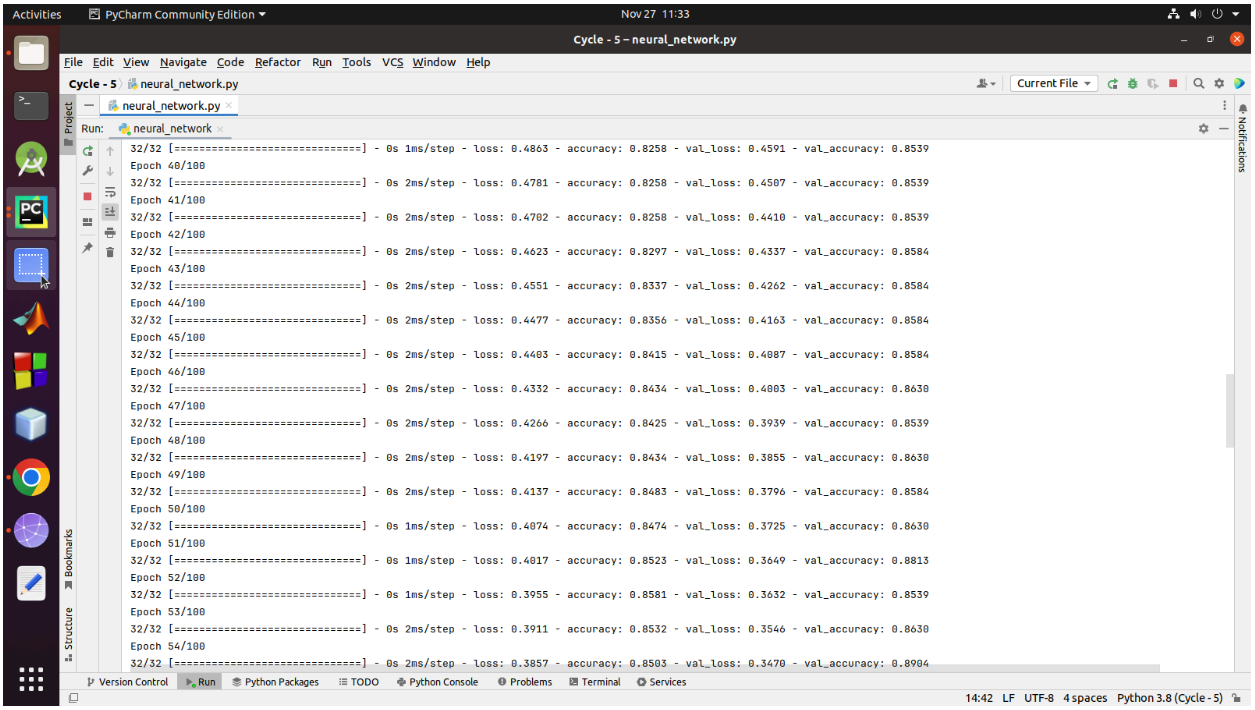
OUTPUT:

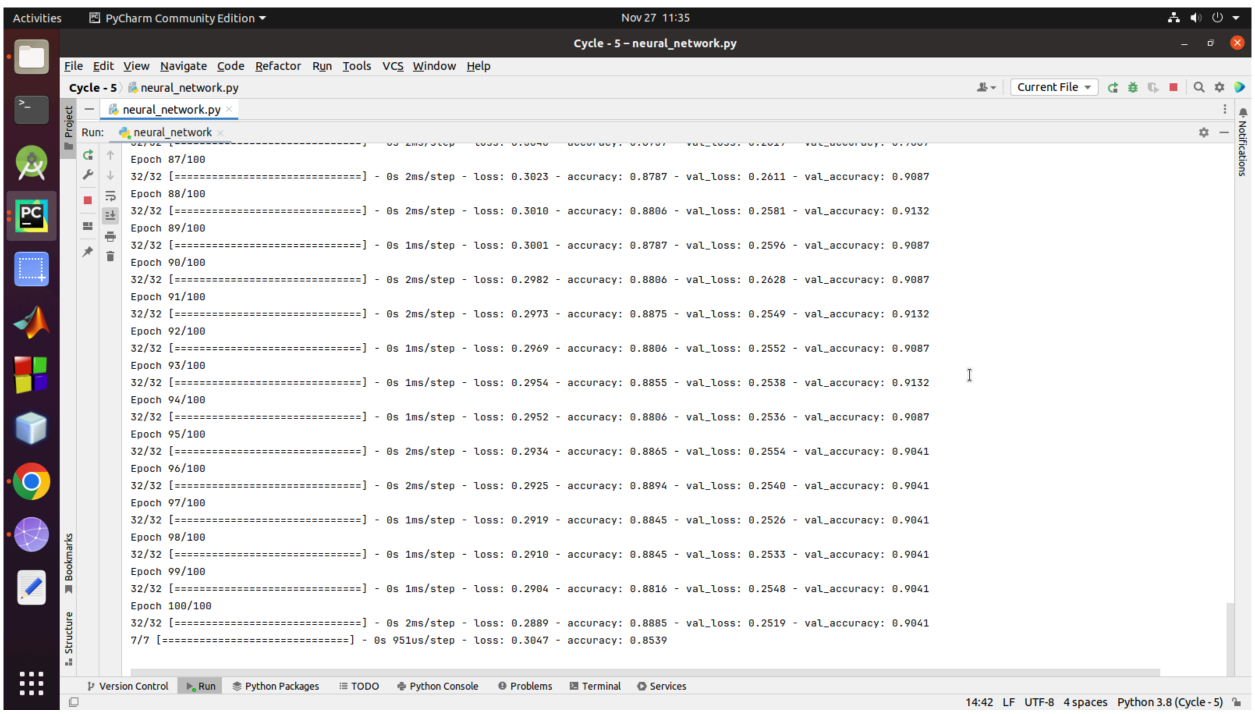





Activities
PyCharm Community Edition
Nov 27 11:32
Cycle - 5 – neural_network.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Cycle - 5 neural_network.py
Current File
neural_network.py
Run: neural_network
Epoch 8/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6741 - accuracy: 0.7818 - val_loss: 0.6708 - val_accuracy: 0.8311
Epoch 9/100
32/32 [==============================] - 0s 1ms/step - loss: 0.6707 - accuracy: 0.7593 - val_loss: 0.6668 - val_accuracy: 0.8311
Epoch 10/100
32/32 [==============================] - 0s 1ms/step - loss: 0.6672 - accuracy: 0.7720 - val_loss: 0.6627 - val_accuracy: 0.8128
Epoch 11/100
32/32 [==============================] - 0s 1ms/step - loss: 0.6634 - accuracy: 0.7622 - val_loss: 0.6581 - val_accuracy: 0.8082
Epoch 12/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6594 - accuracy: 0.7339 - val_loss: 0.6536 - val_accuracy: 0.8082
Epoch 13/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6555 - accuracy: 0.7524 - val_loss: 0.6490 - val_accuracy: 0.7854
Epoch 14/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6515 - accuracy: 0.7436 - val_loss: 0.6447 - val_accuracy: 0.7991
Epoch 15/100
32/32 [==============================] - 0s 1ms/step - loss: 0.6476 - accuracy: 0.7534 - val_loss: 0.6401 - val_accuracy: 0.8037
Epoch 16/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6434 - accuracy: 0.7436 - val_loss: 0.6356 - val_accuracy: 0.8265
Epoch 17/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6390 - accuracy: 0.7544 - val_loss: 0.6306 - val_accuracy: 0.8265
Epoch 18/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6345 - accuracy: 0.7593 - val_loss: 0.6254 - val_accuracy: 0.8265
Epoch 19/100
32/32 [==============================] - 0s 1ms/step - loss: 0.6296 - accuracy: 0.7681 - val_loss: 0.6200 - val_accuracy: 0.8356
Epoch 20/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6245 - accuracy: 0.7779 - val_loss: 0.6141 - val_accuracy: 0.8356
Epoch 21/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6192 - accuracy: 0.7769 - val_loss: 0.6079 - val_accuracy: 0.8356
Epoch 22/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6135 - accuracy: 0.7808 - val_loss: 0.6017 - val_accuracy: 0.8356
Epoch 23/100
Version Control Run Python Packages TODO Python Console Problems Terminal Services
14:42 LF UTF-8 4 spaces Python 3.8 (Cycle - 5)


Activities
PyCharm Community Edition
Nov 27 11:33
Cycle - 5 – neural_network.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Cycle - 5 neural_network.py
Current File
neural_network.py
Run: neural_network
Epoch 23/100
32/32 [==============================] - 0s 1ms/step - loss: 0.6078 - accuracy: 0.7896 - val_loss: 0.5948 - val_accuracy: 0.8402
Epoch 24/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6014 - accuracy: 0.7857 - val_loss: 0.5881 - val_accuracy: 0.8402
Epoch 25/100
32/32 [==============================] - 0s 2ms/step - loss: 0.5951 - accuracy: 0.8014 - val_loss: 0.5806 - val_accuracy: 0.8402
Epoch 26/100
32/32 [==============================] - 0s 1ms/step - loss: 0.5884 - accuracy: 0.7984 - val_loss: 0.5730 - val_accuracy: 0.8402
Epoch 27/100
32/32 [==============================] - 0s 1ms/step - loss: 0.5816 - accuracy: 0.7994 - val_loss: 0.5653 - val_accuracy: 0.8447
Epoch 28/100
32/32 [==============================] - 0s 2ms/step - loss: 0.5744 - accuracy: 0.8072 - val_loss: 0.5572 - val_accuracy: 0.8447
Epoch 29/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5671 - accuracy: 0.8112 - val_loss: 0.5486 - val_accuracy: 0.8447
Epoch 30/100
32/32 [==============================] - 0s 1ms/step - loss: 0.5595 - accuracy: 0.8131 - val_loss: 0.5401 - val_accuracy: 0.8447
Epoch 31/100
32/32 [==============================] - 0s 1ms/step - loss: 0.5514 - accuracy: 0.8151 - val_loss: 0.5314 - val_accuracy: 0.8447
Epoch 32/100
32/32 [==============================] - 0s 2ms/step - loss: 0.5435 - accuracy: 0.8141 - val_loss: 0.5223 - val_accuracy: 0.8447
Epoch 33/100
32/32 [==============================] - 0s 2ms/step - loss: 0.5356 - accuracy: 0.8190 - val_loss: 0.5138 - val_accuracy: 0.8447
Epoch 34/100
32/32 [==============================] - 0s 2ms/step - loss: 0.5274 - accuracy: 0.8190 - val_loss: 0.5045 - val_accuracy: 0.8447
Epoch 35/100
32/32 [==============================] - 0s 1ms/step - loss: 0.5191 - accuracy: 0.8160 - val_loss: 0.4949 - val_accuracy: 0.8493
Epoch 36/100
32/32 [==============================] - 0s 2ms/step - loss: 0.5108 - accuracy: 0.8258 - val_loss: 0.4861 - val_accuracy: 0.8493
Epoch 37/100
32/32 [==============================] - 0s 2ms/step - loss: 0.5026 - accuracy: 0.8209 - val_loss: 0.4768 - val_accuracy: 0.8539
Epoch 38/100
Version Control Run Python Packages TODO Python Console Problems Terminal Services
14:42 LF UTF-8 4 spaces Python 3.8 (Cycle - 5)


Activities
PyCharm Community Edition
Nov 27 11:33
Cycle - 5 – neural_network.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Cycle - 5 neural_network.py
Current File
neural_network.py
Run: neural_network
32/32 [==============================] - 0s 1ms/step - loss: 0.4863 - accuracy: 0.8258 - val_loss: 0.4591 - val_accuracy: 0.8539
Epoch 40/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4781 - accuracy: 0.8258 - val_loss: 0.4507 - val_accuracy: 0.8539
Epoch 41/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4702 - accuracy: 0.8258 - val_loss: 0.4410 - val_accuracy: 0.8539
Epoch 42/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4623 - accuracy: 0.8297 - val_loss: 0.4337 - val_accuracy: 0.8584
Epoch 43/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4551 - accuracy: 0.8337 - val_loss: 0.4262 - val_accuracy: 0.8584
Epoch 44/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4477 - accuracy: 0.8356 - val_loss: 0.4163 - val_accuracy: 0.8584
Epoch 45/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4403 - accuracy: 0.8415 - val_loss: 0.4087 - val_accuracy: 0.8584
Epoch 46/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4332 - accuracy: 0.8434 - val_loss: 0.4003 - val_accuracy: 0.8630
Epoch 47/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4266 - accuracy: 0.8425 - val_loss: 0.3939 - val_accuracy: 0.8539
Epoch 48/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4197 - accuracy: 0.8434 - val_loss: 0.3855 - val_accuracy: 0.8630
Epoch 49/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4137 - accuracy: 0.8483 - val_loss: 0.3796 - val_accuracy: 0.8584
Epoch 50/100
32/32 [==============================] - 0s 1ms/step - loss: 0.4074 - accuracy: 0.8474 - val_loss: 0.3725 - val_accuracy: 0.8630
Epoch 51/100
32/32 [==============================] - 0s 1ms/step - loss: 0.4017 - accuracy: 0.8523 - val_loss: 0.3649 - val_accuracy: 0.8813
Epoch 52/100
32/32 [==============================] - 0s 1ms/step - loss: 0.3955 - accuracy: 0.8581 - val_loss: 0.3632 - val_accuracy: 0.8539
Epoch 53/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3911 - accuracy: 0.8532 - val_loss: 0.3546 - val_accuracy: 0.8630
Epoch 54/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3857 - accuracy: 0.8503 - val_loss: 0.3470 - val_accuracy: 0.8904
Version Control Run Python Packages TODO Python Console Problems Terminal Services
14:42 LF UTF-8 4 spaces Python 3.8 (Cycle - 5)


Activities
PyCharm Community Edition
Nov 27 11:35
Cycle - 5 – neural_network.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Cycle - 5 neural_network.py
Current File
neural_network.py
Run: neural_network
Epoch 87/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3023 - accuracy: 0.8787 - val_loss: 0.2611 - val_accuracy: 0.9087
Epoch 88/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3010 - accuracy: 0.8806 - val_loss: 0.2581 - val_accuracy: 0.9132
Epoch 89/100
32/32 [==============================] - 0s 1ms/step - loss: 0.3001 - accuracy: 0.8787 - val_loss: 0.2596 - val_accuracy: 0.9087
Epoch 90/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2982 - accuracy: 0.8806 - val_loss: 0.2628 - val_accuracy: 0.9087
Epoch 91/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2973 - accuracy: 0.8875 - val_loss: 0.2549 - val_accuracy: 0.9132
Epoch 92/100
32/32 [==============================] - 0s 1ms/step - loss: 0.2969 - accuracy: 0.8806 - val_loss: 0.2552 - val_accuracy: 0.9087
Epoch 93/100
32/32 [==============================] - 0s 1ms/step - loss: 0.2954 - accuracy: 0.8855 - val_loss: 0.2538 - val_accuracy: 0.9132
Epoch 94/100
32/32 [==============================] - 0s 1ms/step - loss: 0.2952 - accuracy: 0.8806 - val_loss: 0.2536 - val_accuracy: 0.9087
Epoch 95/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2934 - accuracy: 0.8865 - val_loss: 0.2554 - val_accuracy: 0.9041
Epoch 96/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2925 - accuracy: 0.8894 - val_loss: 0.2540 - val_accuracy: 0.9041
Epoch 97/100
32/32 [==============================] - 0s 1ms/step - loss: 0.2919 - accuracy: 0.8845 - val_loss: 0.2526 - val_accuracy: 0.9041
Epoch 98/100
32/32 [==============================] - 0s 1ms/step - loss: 0.2910 - accuracy: 0.8845 - val_loss: 0.2533 - val_accuracy: 0.9041
Epoch 99/100
32/32 [==============================] - 0s 1ms/step - loss: 0.2904 - accuracy: 0.8816 - val_loss: 0.2548 - val_accuracy: 0.9041
Epoch 100/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2889 - accuracy: 0.8885 - val_loss: 0.2519 - val_accuracy: 0.9041
7/7 [==============================] - 0s 951us/step - loss: 0.3047 - accuracy: 0.8539
Version Control Run Python Packages TODO Python Console Problems Terminal Services
14:42 LF UTF-8 4 spaces Python 3.8 (Cycle - 5)