REGRESSION

Accuracy using ANN

In [90]:
```
# importing requered libraries
import numpy as np
import pandas as pd
import tensorflow as tf
```

In [91]:
```
##reading the dataset into a variable
dataset = pd.read_excel('Folds5x2_pp.xlsx')
```

In [92]:
```
#printing first five coloumns of dataset
dataset.head()
```

Out[92]:

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

In [93]:
```
#checking the dataset for any Null values
dataset.isnull().sum()
```

Out[93]:
```
AT    0
V     0
AP    0
RH    0
PE    0
dtype: int64
```

In [94]:
```
#fetching information of each row in the dataset
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9568 entries, 0 to 9567
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   AT      9568 non-null   float64
 1   V       9568 non-null   float64
 2   AP      9568 non-null   float64
 3   RH      9568 non-null   float64
 4   PE      9568 non-null   float64
dtypes: float64(5)
memory usage: 373.9 KB
```

In [95]:
```
#getting the column and row size of data
dataset.shape
```

Out[95]:
```
(9568, 5)
```

In [96]:
```
#splitting the rows into input and output
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, -1].values
```

In [97]:
```
# Separating the data into train and test.
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, shuffle=True)

X_train, X_val, y_train, y_val = train_test_split(X_train,y_train, test_size = 0.2, shuf

X_train.shape, y_train.shape,X_test.shape,y_test.shape
```

Out[97]:
```
((6123, 4), (6123,), (1914, 4), (1914,))
```

In [98]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

In [99]:
```
#designing the model using tenserflow
ann = Sequential()

ann.add(Dense(units=4, activation='relu'))#input layer

ann.add(Dense(units=6, activation='relu'))#hidden layer

ann.add(Dense(units=1))#output layer
```

In [118…
```
#using adam optimizer to compile the model
ann.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

In [102…
```
model = ann.fit(X_train, y_train, batch_size = 32, epochs = 25,validation_data=(X_val, y
              shuffle=True)
```

```
Epoch 1/25
192/192 [==============================] - 1s 3ms/step - loss: 1343.2537 - val_loss: 15
9.7437
Epoch 2/25
192/192 [==============================] - 0s 2ms/step - loss: 147.2841 - val_loss: 149.
7636
Epoch 3/25
192/192 [==============================] - 0s 2ms/step - loss: 136.6754 - val_loss: 137.
9041
Epoch 4/25
192/192 [==============================] - 0s 2ms/step - loss: 125.2263 - val_loss: 124.
8189
Epoch 5/25
192/192 [==============================] - 0s 2ms/step - loss: 112.7543 - val_loss: 111.
8021
Epoch 6/25
192/192 [==============================] - 0s 3ms/step - loss: 100.1540 - val_loss: 98.4
151
Epoch 7/25
192/192 [==============================] - 1s 3ms/step - loss: 88.3959 - val_loss: 86.20
34
Epoch 8/25
192/192 [==============================] - 0s 3ms/step - loss: 77.8604 - val_loss: 75.36
09
Epoch 9/25
192/192 [==============================] - 1s 3ms/step - loss: 67.8400 - val_loss: 65.58
31
Epoch 10/25
192/192 [==============================] - 1s 3ms/step - loss: 59.5815 - val_loss: 57.90
41
Epoch 11/25
192/192 [==============================] - 0s 3ms/step - loss: 52.9429 - val_loss: 50.76
07
Epoch 12/25
192/192 [==============================] - 0s 3ms/step - loss: 47.1470 - val_loss: 45.25
06
Epoch 13/25
192/192 [==============================] - 1s 3ms/step - loss: 42.9707 - val_loss: 41.66
32
Epoch 14/25
192/192 [==============================] - 1s 3ms/step - loss: 39.1026 - val_loss: 37.50
48
Epoch 15/25
192/192 [==============================] - 1s 4ms/step - loss: 36.5159 - val_loss: 34.83
86
Epoch 16/25
192/192 [==============================] - 1s 3ms/step - loss: 34.4723 - val_loss: 32.75
09
Epoch 17/25
192/192 [==============================] - 0s 2ms/step - loss: 32.8541 - val_loss: 31.59
13
Epoch 18/25
192/192 [==============================] - 0s 2ms/step - loss: 31.6036 - val_loss: 30.38
34
Epoch 19/25
192/192 [==============================] - 0s 3ms/step - loss: 31.4974 - val_loss: 29.50
35
Epoch 20/25
192/192 [==============================] - 0s 3ms/step - loss: 30.1345 - val_loss: 28.41
50
Epoch 21/25
192/192 [==============================] - 0s 2ms/step - loss: 30.3752 - val_loss: 30.68
11
Epoch 22/25
```

```
192/192 [==============================] - 0s 2ms/step - loss: 29.7382 - val_loss: 28.26
50
Epoch 23/25
192/192 [==============================] - 0s 2ms/step - loss: 29.5514 - val_loss: 27.88
38
Epoch 24/25
192/192 [==============================] - 0s 3ms/step - loss: 28.8701 - val_loss: 27.10
42
Epoch 25/25
192/192 [==============================] - 1s 3ms/step - loss: 28.9305 - val_loss: 26.62
37
```

In [103…
```
#Predicting the results of the Test set
y_pred = ann.predict(X_test)
```

```
60/60 [==============================] - 0s 2ms/step
```

In [104…
```
#calculating accuracy
from sklearn.metrics import r2_score

Accuracy=r2_score(y_test,y_pred)
```

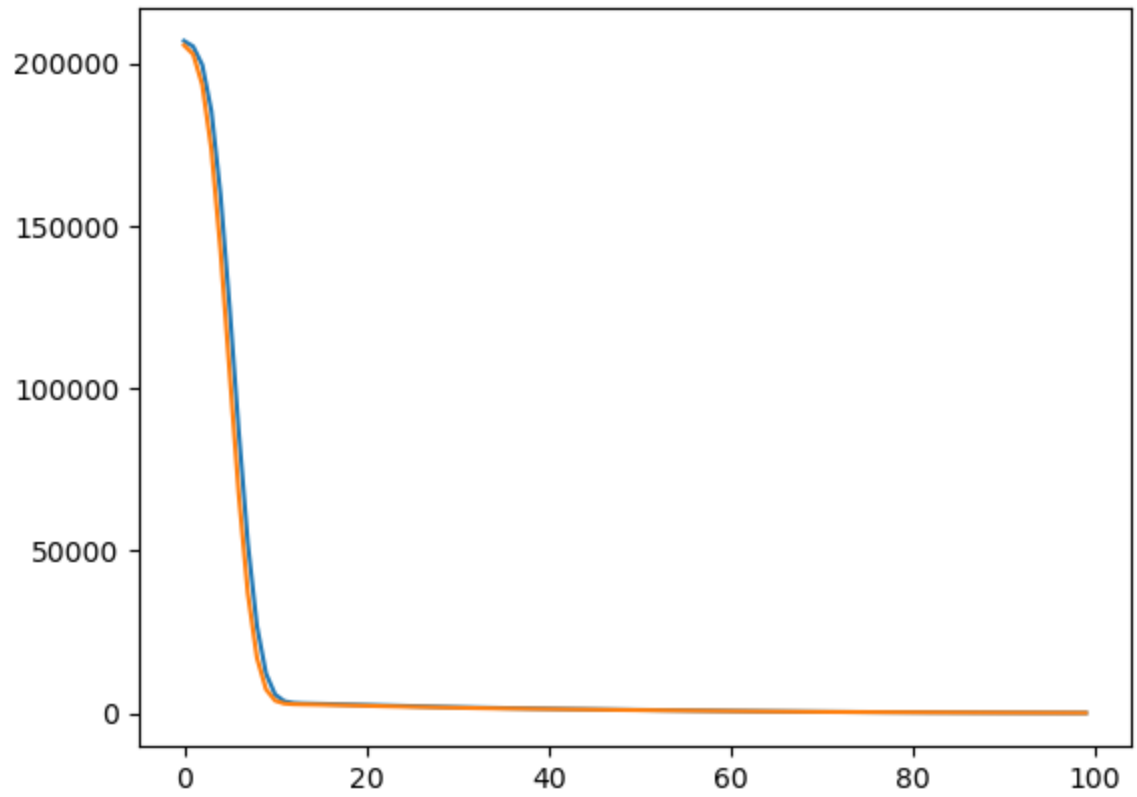
In [105…
```
Accuracy
```

Out[105]:
```
0.9051268626316156
```

In [106…
```
import matplotlib.pyplot as plt
```

In [107…
```
#plotting training and validation loss

plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.show()
```



Accuracy using Linear Regression

In [108…
```
# importing requered libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [109…
```
data = pd.read_excel('Folds5x2_pp.xlsx')
```

In [110…
```
data
```

Out[110]:

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90
...	...	...	...	...	...
9563	16.65	49.69	1014.01	91.00	460.03
9564	13.19	39.18	1023.67	66.78	469.62
9565	31.32	74.33	1012.92	36.48	429.57
9566	24.48	69.45	1013.86	62.39	435.74
9567	21.60	62.52	1017.23	67.87	453.28

9568 rows × 5 columns

In [111…
```
data.isnull().sum()
```

Out[111]:
```
AT    0
V     0
AP    0
RH    0
PE    0
dtype: int64
```

In [112…
```
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, -1].values
```

In [113…
```
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)


X_train.shape, y_train.shape,X_test.shape,y_test.shape
```

Out[113]:
```
((7654, 4), (7654,), (1914, 4), (1914,))
```

In [114…
```
#creating model using linear regression algorithm
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()
lin_model.fit(X_train, y_train)
```

Out[114]:

```
▼ LinearRegression
LinearRegression()
```

In [115…

```
#predicting and calculating the accuracy
from sklearn.metrics import r2_score

y_test_predict = lin_model.predict(X_test)

b=r2_score(y_test,y_test_predict)
```

In [116…

```
b
```

Out[116]:

```
0.9235606731016073
```

Accuracy score

ANN: 90.51% ~ 91%

Linear Regression: 92.35% ~ 92%

DATASET

https://www.kaggle.com/datasets/gauravduttakiit/combined-cycle-power-plant

Classification

Accuracy using ANN

In [1]:
```
# importing requered libraries
import numpy as np
import pandas as pd
```

In [2]:
```
#reading the datasheet into a variable
data = pd.read_csv("Employee.csv")
```

In [3]:
```
data.shape
```

Out[3]:
```
(4653, 9)
```

In [4]:
```
data.head()
```

Out[4]:

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	Lea
0	Bachelors	2017	Bangalore	3	34	Male	No	0	
1	Bachelors	2013	Pune	1	28	Female	No	3	
2	Bachelors	2014	New Delhi	3	38	Female	No	2	
3	Masters	2016	Bangalore	3	27	Male	No	5	
4	Masters	2017	Pune	3	24	Male	Yes	2	

In [7]:
```
#checking the data information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4653 entries, 0 to 4652
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Education                  4653 non-null   object
 1   JoiningYear                4653 non-null   int64
 2   City                       4653 non-null   object
 3   PaymentTier                4653 non-null   int64
 4   Age                        4653 non-null   int64
 5   Gender                     4653 non-null   object
 6   EverBenched                4653 non-null   object
 7   ExperienceInCurrentDomain  4653 non-null   int64
 8   LeaveOrNot                 4653 non-null   int64
dtypes: int64(5), object(4)
memory usage: 327.3+ KB
```

In [8]:
```
#Cheaking for duplicate values
data.duplicated().sum()
```

Out[8]:
```
1889
```

In [9]:
```
#Removing duplicate values
print(data[~data.duplicated()])
```

```
      Education  JoiningYear       City  PaymentTier  Age  Gender EverBenched  \
0     Bachelors         2017  Bangalore            3   34    Male          No
1     Bachelors         2013       Pune            1   28  Female          No
2     Bachelors         2014  New Delhi            3   38  Female          No
3       Masters         2016  Bangalore            3   27    Male          No
4       Masters         2017       Pune            3   24    Male         Yes
...         ...          ...        ...          ...  ...     ...         ...
4645    Masters         2017       Pune            2   31  Female          No
4647  Bachelors         2016       Pune            3   30    Male          No
4649    Masters         2013       Pune            2   37    Male          No
4650    Masters         2018  New Delhi            3   27    Male          No
4651  Bachelors         2012  Bangalore            3   30    Male         Yes

      ExperienceInCurrentDomain  LeaveOrNot
0                             0           0
1                             3           1
2                             2           0
3                             5           1
4                             2           1
...                         ...         ...
4645                          2           0
4647                          2           0
4649                          2           1
4650                          5           1
4651                          2           0

[2764 rows x 9 columns]
```

In [10]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
```

In [13]:
```
categorical_columns = ['Gender', 'EverBenched', 'Education','City']

# Apply label encoding to each categorical column
for column in categorical_columns:
    data[column] = label_encoder.fit_transform(data[column])
```

In [14]:
```
#spliting the data into output
x = data.drop(columns=['LeaveOrNot'])
y = data['LeaveOrNot']
```

In [15]:
```
from sklearn.model_selection import train_test_split
# Separating the data into train and test.
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=0.2,random_state=108)
```

In [16]:
```
train_x.shape,test_x.shape,train_y.shape,test_y.shape
```

Out[16]:
```
((3722, 8), (931, 8), (3722,), (931,))
```

In [17]:
```
from sklearn.preprocessing import StandardScaler

scal = StandardScaler()

train_x_scal = scal.fit_transform(train_x)
test_x_scal = scal.fit_transform(test_x)
```

In [18]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

In [23]:
```
#creating the model
model = Sequential()

#input layer
model.add(Dense(5,activation='sigmoid',input_dim=8))
#hidden layer
model.add(Dense(6, activation='relu'))
#output layer
model.add(Dense(1,activation='relu'))
```

In [24]:
```
#printing the summary for model
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 5)                 45

 dense_4 (Dense)             (None, 6)                 36

 dense_5 (Dense)             (None, 1)                 7

=================================================================
Total params: 88 (352.00 Byte)
Trainable params: 88 (352.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [25]:
```
#binary_crossentropy or can say log loss
model.compile(loss='binary_crossentropy',optimizer='Adam')
```

In [26]:
```
hist = model.fit(train_x_scal,train_y,epochs=25)
```

```
Epoch 1/25
117/117 [==============================] - 1s 2ms/step - loss: 2.3144
Epoch 2/25
117/117 [==============================] - 0s 2ms/step - loss: 0.7991
Epoch 3/25
117/117 [==============================] - 0s 2ms/step - loss: 0.7136
Epoch 4/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6942
Epoch 5/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6792
Epoch 6/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6658
Epoch 7/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6535
Epoch 8/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6425
Epoch 9/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6327
Epoch 10/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6240
Epoch 11/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6163
Epoch 12/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6096
Epoch 13/25
117/117 [==============================] - 0s 2ms/step - loss: 0.6038
Epoch 14/25
117/117 [==============================] - 0s 2ms/step - loss: 0.5987
Epoch 15/25
117/117 [==============================] - 0s 2ms/step - loss: 0.5943
Epoch 16/25
117/117 [==============================] - 0s 2ms/step - loss: 0.5904
Epoch 17/25
117/117 [==============================] - 0s 2ms/step - loss: 0.5871
Epoch 18/25
117/117 [==============================] - 0s 2ms/step - loss: 0.5839
Epoch 19/25
117/117 [==============================] - 0s 3ms/step - loss: 0.5811
Epoch 20/25
117/117 [==============================] - 0s 2ms/step - loss: 0.5782
Epoch 21/25
117/117 [==============================] - 0s 3ms/step - loss: 0.5753
Epoch 22/25
117/117 [==============================] - 0s 2ms/step - loss: 0.5726
Epoch 23/25
117/117 [==============================] - 0s 3ms/step - loss: 0.5688
Epoch 24/25
117/117 [==============================] - 0s 3ms/step - loss: 0.5669
Epoch 25/25
117/117 [==============================] - 0s 3ms/step - loss: 0.5686
```

In [28]:
```
y_pred = np.where(model.predict(test_x_scal)>0.5,1,0)
```

```
30/30 [==============================] - 0s 2ms/step
```

In [29]:
```
from sklearn.metrics import accuracy_score
#Claculating acuracy score
accuracy_score(test_y,y_pred)
```

Out[29]:
```
0.6992481203007519
```

Accuracy using Random Forest Classifier

In [30]:
```
# importing data and requered libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [31]:
```
df = pd.read_csv("Employee.csv")
```

In [32]:
```
df.head()
```

Out[32]:

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	Lea
0	Bachelors	2017	Bangalore	3	34	Male	No	0	
1	Bachelors	2013	Pune	1	28	Female	No	3	
2	Bachelors	2014	New Delhi	3	38	Female	No	2	
3	Masters	2016	Bangalore	3	27	Male	No	5	
4	Masters	2017	Pune	3	24	Male	Yes	2	

In [33]:
```
df.shape
```

Out[33]:
```
(4653, 9)
```

In [36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4653 entries, 0 to 4652
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Education                  4653 non-null   object
 1   JoiningYear                4653 non-null   int64
 2   City                       4653 non-null   object
 3   PaymentTier                4653 non-null   int64
 4   Age                        4653 non-null   int64
 5   Gender                     4653 non-null   object
 6   EverBenched                4653 non-null   object
 7   ExperienceInCurrentDomain  4653 non-null   int64
 8   LeaveOrNot                 4653 non-null   int64
dtypes: int64(5), object(4)
memory usage: 327.3+ KB
```

In [37]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
```

In [38]:
```
categorical_columns = ['Gender', 'EverBenched', 'Education','City']

# Apply label encoding to each categorical column
for column in categorical_columns:
    df[column] = label_encoder.fit_transform(df[column])
```

In [39]:
```
X = df.drop('LeaveOrNot', axis=1)
y = df['LeaveOrNot']
```

In [40]:
```
# Separating the data into train and test.(%20 Test set / %80 Train set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [41]:
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [42]:
```
model.fit(X_train, y_train)
```

Out[42]:
```
▼         RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [43]:
```
# Make predictions on the test set

predictions= model.predict(X_test)
```

In [44]:
```
# Calculating accuracy score
print("Accuracy Score :",accuracy_score(y_test,predictions))
print("Classification Report \n",classification_report(y_test,predictions))
```

```
Accuracy Score : 0.849624060150376
Classification Report
               precision    recall  f1-score   support

           0       0.86      0.92      0.89       610
           1       0.82      0.72      0.77       321

    accuracy                           0.85       931
   macro avg       0.84      0.82      0.83       931
weighted avg       0.85      0.85      0.85       931
```

Accuracy score

ANN: 69.95% ~ 70%

RandomForestClassifier: 84.96% ~ 85%

DATASET

https://www.kaggle.com/datasets/tawfikelmetwally/employee-dataset