MY 2nd ML PROJECT || Kaggle UCS 654 lab exam set

ANANDA JANA | March 17 , 2025

In [20]:
```
import pandas as pd
import numpy as np
```

In [21]:
```
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
```

In [22]:
```
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
```

now check data is there or not

In [23]:
```
train_df.head(4)
```

Out[23]:

	target	f1	f2	f3	f4	f5	f6
0	27.4	47.2	40.2	-16.0	13	7.9	31.7
1	15.6	40.6	21.9	-11.5	20	5.4	16.5
2	23.6	47.7	27.9	-12.6	46	6.7	22.4
3	38.9	82.7	95.5	-28.5	26	13.8	55.4

In [24]:
```
test_df.head(4)
```

Out[24]:

	id	f1	f2	f3	f4	f5	f6
0	1	129.3	663.7	-75.3	52	29.3	298.0
1	2	143.1	687.3	-82.6	63	30.7	306.2
2	3	52.3	32.0	-10.8	39	7.1	24.9
3	4	25.1	0.5	-5.6	8	3.3	0.5

splitting NOW the features colums in one var and target in other

In [25]:

```
x=train_df.drop(columns=['target'])
y=train_df['target']
x.head(2)
```

Out[25]:

	f1	f2	f3	f4	f5	f6
0	47.2	40.2	-16.0	13	7.9	31.7
1	40.6	21.9	-11.5	20	5.4	16.5

In [26]:

```
y.head(3)
```

Out[26]:

```
0    27.4
1    15.6
2    23.6
Name: target, dtype: float64
```

NOW WE HAVE TO SCALE OUR FEATURES

In [29]:

```
scl=StandardScaler()
x_scaled= scl.fit_transform(x)
x_scaled[:5]
```

Out[29]:
```
array([[-1.18232172, -0.56464608,  0.06984499, -0.94299693, -1.0238104 ,
        -0.68270123],
       [-1.3068134 , -0.57905385,  0.07927569, -0.79813107, -1.21548566,
        -0.72280679],
       [-1.17289053, -0.57432999,  0.07697041, -0.26005787, -1.11581452,
        -0.7072395 ],
       [-0.51270733, -0.52110787,  0.04364859, -0.67396033, -0.5714568 ,
        -0.62016822],
       [-0.73339714, -0.48489163,  0.05349844,  0.54705194, -0.70946298,
        -0.55288586]])
```

Split into training & validation sets

In [34]:
```
x_train, x_val, y_train, y_val = train_test_split(x_scaled, y, test_size=0.2, random_state=42)
x_train[:3]
```

Out[34]:
```
array([[-1.03519517, -0.56133938,  0.08241926, -0.79813107, -0.95480731,
        -0.68929754],
       [ 2.22422359,  1.59714831, -0.08230368,  0.17453972,  1.95865658,
         1.68115216],
       [-0.92956586, -0.49465427,  0.09855624,  0.07106411, -0.79380009,
        -0.55367742]])
```

In [35]:
```
x_val[:3]
```

Out[35]:
```
array([[-0.79375675, -0.49544158,  0.07110241, -0.57048471, -0.89347123,
        -0.55420512],
       [-0.45989267, -0.38655668,  0.05643243, -0.65326521, -0.43345061,
        -0.4064478 ],
       [ 1.01514522,  0.40878337, -0.06931027,  0.1538446 ,  1.1536205 ,
         0.47983232]])
```

In [38]:
```
!pip install --upgrade xgboost
```

```
Requirement already satisfied: xgboost in w:\sw\python\python312\lib\site-packages (3.0.0)
Requirement already satisfied: numpy in w:\sw\python\python312\lib\site-packages (from xgboost) (2.0.2)
Requirement already satisfied: scipy in w:\sw\python\python312\lib\site-packages (from xgboost) (1.15.1)
```

In [40]:
```
xgb_model = xgb.XGBRegressor(n_estimators=500, learning_rate=0.05, max_depth=6,
                             subsample=0.8, colsample_bytree=0.8, random_state=42)
```

```
xgb_model.fit(x_train, y_train, eval_set=[(x_val, y_val)], verbose=True)
```

```
[0]	validation_0-rmse:16.77189
[1]	validation_0-rmse:16.51275
[2]	validation_0-rmse:16.37119
[3]	validation_0-rmse:16.13329
[4]	validation_0-rmse:15.94156
[5]	validation_0-rmse:15.80762
[6]	validation_0-rmse:15.69454
[7]	validation_0-rmse:15.60661
[8]	validation_0-rmse:15.42499
[9]	validation_0-rmse:15.34093
[10]	validation_0-rmse:15.24054
[11]	validation_0-rmse:15.04609
[12]	validation_0-rmse:14.86462
[13]	validation_0-rmse:14.71400
[14]	validation_0-rmse:14.56916
[15]	validation_0-rmse:14.43217
[16]	validation_0-rmse:14.30796
[17]	validation_0-rmse:14.20340
[18]	validation_0-rmse:14.10152
[19]	validation_0-rmse:14.04962
[20]	validation_0-rmse:13.96033
[21]	validation_0-rmse:13.83681
[22]	validation_0-rmse:13.79949
[23]	validation_0-rmse:13.68870
[24]	validation_0-rmse:13.58628
[25]	validation_0-rmse:13.51710
[26]	validation_0-rmse:13.43352
...
[490]	validation_0-rmse:8.63627
[491]	validation_0-rmse:8.63453
[492]	validation_0-rmse:8.63145
[493]	validation_0-rmse:8.62964
[494]	validation_0-rmse:8.62981
[495]	validation_0-rmse:8.62445
[496]	validation_0-rmse:8.62299
[497]	validation_0-rmse:8.61881
[498]	validation_0-rmse:8.61474
[499]	validation_0-rmse:8.61482
```

Out[40]:
```
▸ XGBRegressor
```

Evaluate on validation set

In [44]:
```
y_pred = xgb_model.predict(x_val)
rmse = np.sqrt(mean_squared_error(y_val, y_pred))
print(f"Validation RMSE: {rmse}")
print("Validation Predictions Head:\n", y_pred[:5])
```

```
Validation RMSE: 8.614816142410081
Validation Predictions Head:
 [28.218363   30.69904     0.38946062 22.599026    3.1642566 ]
```

Prepare test data & make predictions

In [46]:
```
x_test = scl.transform(test_df.drop(columns=["id"]))
print("Test Features Scaled Sample:\n", x_test[:5])
test_predictions = xgb_model.predict(x_test)
print("Test Predictions Head:\n", test_predictions[:5])
```

```
Test Features Scaled Sample:
 [[ 0.36627944 -0.07375862 -0.05443071 -0.13588713  0.61692979  0.01993763]
 [ 0.62658025 -0.05517811 -0.06972941  0.09175923  0.72426793  0.04157353]
 [-1.08612359 -0.57110202  0.08074269 -0.40492373 -1.08514648 -0.70064319]
 [-1.59918025 -0.59590227  0.09164039 -1.04647255 -1.37649287 -0.76502317]
 [ 0.05316398 -0.04415578 -0.02068976 -0.38422861  0.21057825  0.038935  ]]
Test Predictions Head:
 [-2.7747908 -1.7713814 33.704803  12.257837  -3.2744658]
```

now submit

In [47]:
```
submission = pd.DataFrame({"id": test_df["id"], "target": test_predictions})
print("Submission Data Head:\n", submission.head())
submission.to_csv("submission.csv", index=False)
print("Submission file saved.")
```

```
Submission Data Head:
    id     target
0   1  -2.774791
1   2  -1.771381
2   3  33.704803
3   4  12.257837
4   5  -3.274466
Submission file saved.
```