Data Preprocessing

Handling Missing Values: Missing values were checked using df.isnull().sum(). The specific handling strategy was not detailed in the snippet.

Dealing with Duplicates: Duplicate entries were identified using df.duplicated().sum() and were presumably removed.

Data Splitting: Data was split into training, validation, and test subsets using sklearn's train_test_split, with a distribution aimed at creating training, validation, and test sets.

Statistical Summary

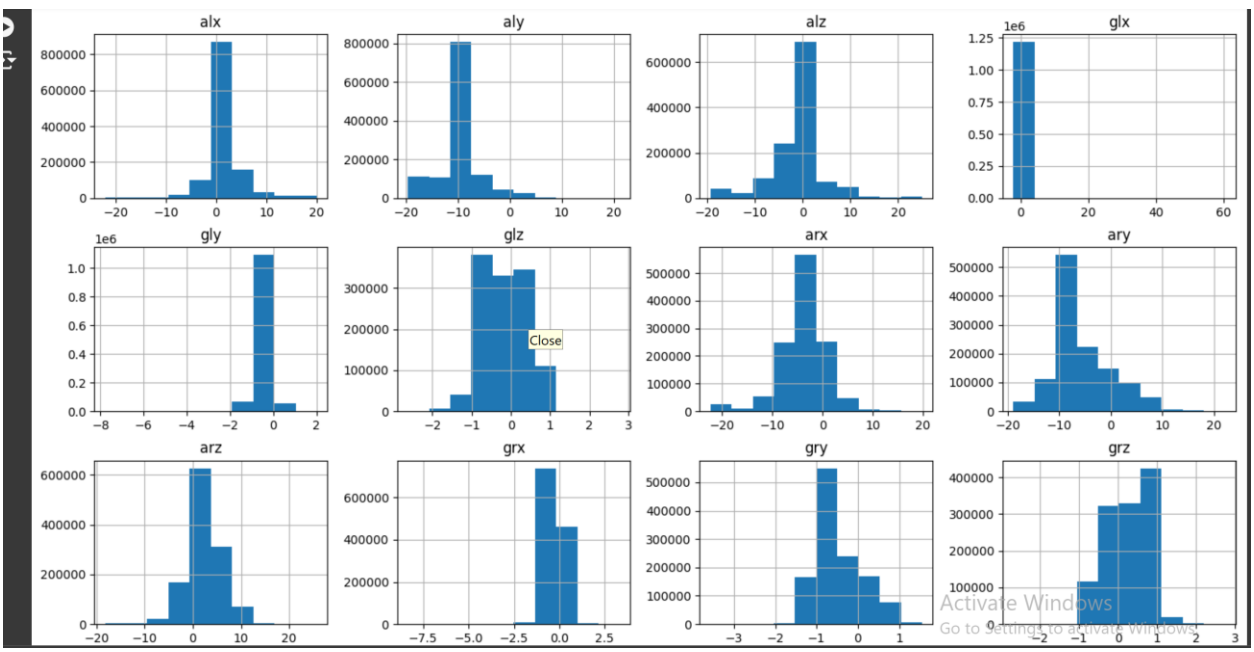
Basic statistical details using df.describe() to understand the central tendency and variability of the data.

```
df.describe()
```

	alx	aly	alz	glx	gly	glz	arx	ary	arz	
count	1.215745e+06	1.215745e+06	1.215745e+06	1.215745e+06	1.215745e+06	1.215745e+06	1.215745e+06	1.215745e+06	1.215745e+06	1.215745e
mean	1.494200e+00	-9.692878e+00	-9.548056e-01	-1.598951e-03	-6.166318e-01	-1.587811e-01	-3.713413e+00	-5.805526e+00	2.393880e+00	-2.761061
std	3.826485e+00	4.171303e+00	5.461803e+00	4.912172e-01	3.546406e-01	5.467979e-01	4.763586e+00	5.757639e+00	3.876503e+00	5.276888
min	-2.214600e+01	-1.961900e+01	-1.937300e+01	-2.146600e+00	-7.789900e+00	-2.626700e+00	-2.236100e+01	-1.897200e+01	-1.823900e+01	-8.339200e
25%	1.413100e-01	-1.020100e+01	-2.649400e+00	-4.359900e-01	-8.180100e-01	-5.933200e-01	-6.076000e+00	-9.404200e+00	1.296500e-01	-7.058800
50%	1.308900e+00	-9.670300e+00	-1.645600e-02	-1.484200e-02	-7.073200e-01	-1.905700e-01	-2.977600e+00	-7.461500e+00	1.928100e+00	-3.549000
75%	2.575800e+00	-9.042200e+00	1.301300e+00	4.489800e-01	-5.403400e-01	3.222000e-01	-1.193700e+00	-2.533900e+00	4.914700e+00	9.607800
max	2.005400e+01	2.116100e+01	2.501500e+01	6.048400e+01	2.011300e+00	2.770100e+00	1.986400e+01	2.219100e+01	2.574100e+01	3.319600e

Data Visualization:

Visualizations included histograms and a heatmap of correlations among numerical features to identify relationships and distribution patterns.



Machine Learning Models

K-Nearest Neighbors (KNN)

Model Building: KNN was used with varying numbers of neighbors to find the optimal.

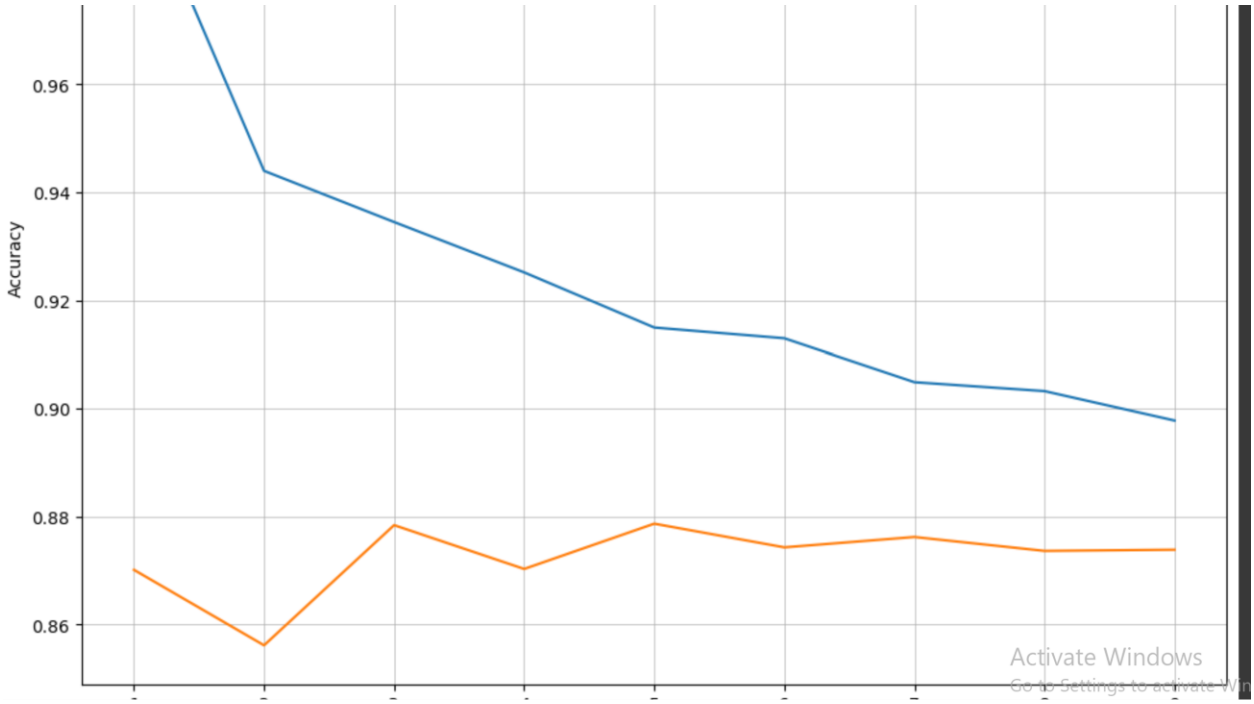
Model Evaluation: Accuracy scores computed for different configurations, with results printed out as lists of accuracies for training, validation, and testing data.

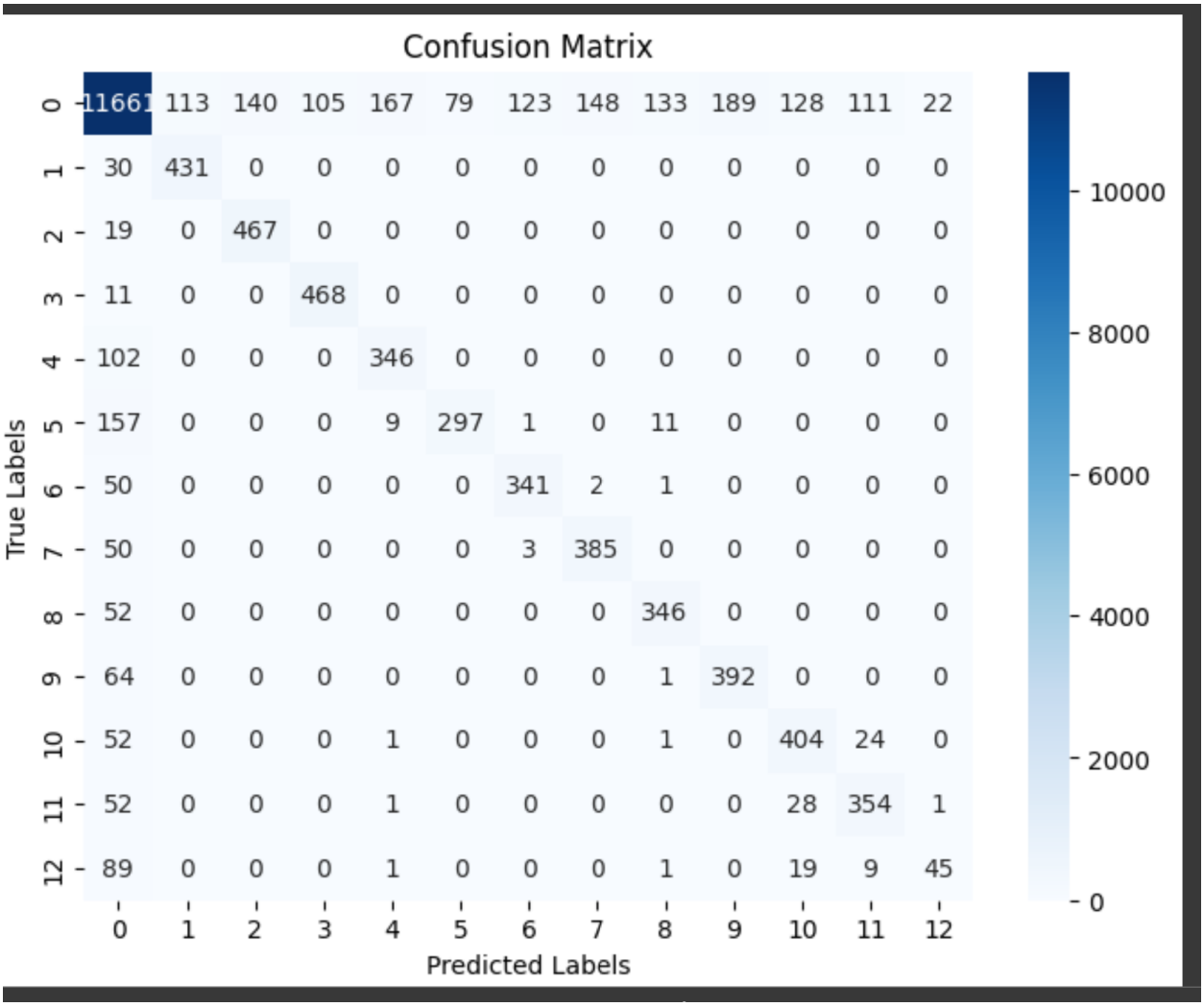
```
print("Training Accuracies:", train_accuracies)

print("Validation Accuracies:", val_accuracies)

print("Test Accuracies:", test_accuracies)
```

```
Training Accuracies: [1.0, 0.9439842070010929, 0.9345013572108436, 0.925171267082643, 0.9149598712118542, 0.9129739956052221, 0.90491298574634
Validation Accuracies: [0.873601667032244, 0.8577538933976749, 0.8748080719456021, 0.8720662425970608, 0.8767821890765519, 0.8733274840973898,
Test Accuracies: [0.8701540823600373, 0.8561715194385041, 0.8784339529527883, 0.8703185831002906, 0.8787081208532105, 0.874321434446455, 0.876
```





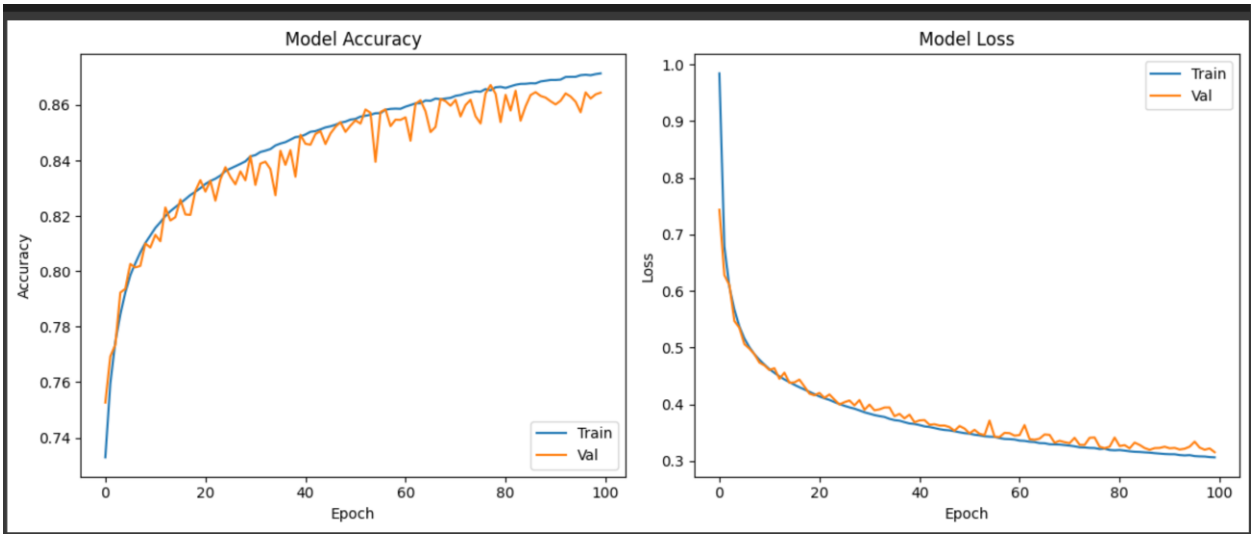
Linear Regression

Evaluation: Performance was assessed using metrics like R^2, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

Support Vector Machine (SVM)

Model Setup: Configured with RBF kernel.

Evaluation: Validation and test accuracies were provided to assess the model's effectiveness.



Neural Network

A multi-layer perceptron model with sigmoid activations.

Training and Evaluation: The model's performance was tracked over epochs, showing training and validation accuracies and losses.

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

hist = model.fit(X_train, y_train,
                 batch_size=1024,
                 epochs=100,
                 validation_data=(X_val, y_val))
```

```
Epoch 1/100
832/832 [==============================] - 9s 9ms/step - loss: 1.3018 - accuracy: 0.7141 - val_loss: 1.2849 - val_accuracy: 0.7178
Epoch 2/100
832/832 [==============================] - 9s 11ms/step - loss: 1.1729 - accuracy: 0.7236 - val_loss: 1.0629 - val_accuracy: 0.7262
Epoch 3/100
832/832 [==============================] - 7s 8ms/step - loss: 1.0226 - accuracy: 0.7261 - val_loss: 0.9865 - val_accuracy: 0.7274
Epoch 4/100
832/832 [==============================] - 10s 12ms/step - loss: 0.9526 - accuracy: 0.7276 - val_loss: 0.9215 - val_accuracy: 0.7295
Epoch 5/100
832/832 [==============================] - 7s 8ms/step - loss: 0.8990 - accuracy: 0.7309 - val_loss: 0.8761 - val_accuracy: 0.7318
Epoch 6/100
832/832 [==============================] - 10s 12ms/step - loss: 0.8580 - accuracy: 0.7327 - val_loss: 0.8372 - val_accuracy: 0.7341
Epoch 7/100
832/832 [==============================] - 8s 9ms/step - loss: 0.8212 - accuracy: 0.7355 - val_loss: 0.8058 - val_accuracy: 0.7370
Epoch 8/100
832/832 [==============================] - 9s 11ms/step - loss: 0.7918 - accuracy: 0.7379 - val_loss: 0.7809 - val_accuracy: 0.7389
Epoch 9/100
832/832 [==============================] - 8s 9ms/step - loss: 0.7676 - accuracy: 0.7422 - val_loss: 0.7575 - val_accuracy: 0.7466
Epoch 10/100
832/832 [==============================] - 9s 11ms/step - loss: 0.7457 - accuracy: 0.7468 - val_loss: 0.7357 - val_accuracy: 0.7482
Epoch 11/100
832/832 [==============================] - 11s 14ms/step - loss: 0.7271 - accuracy: 0.7501 - val_loss: 0.7208 - val_accuracy: 0.7501
```

Logistic Regression

Model Setup and Evaluation: Logistic regression was used to classify activities, with performance evaluated through accuracy scores and classification reports

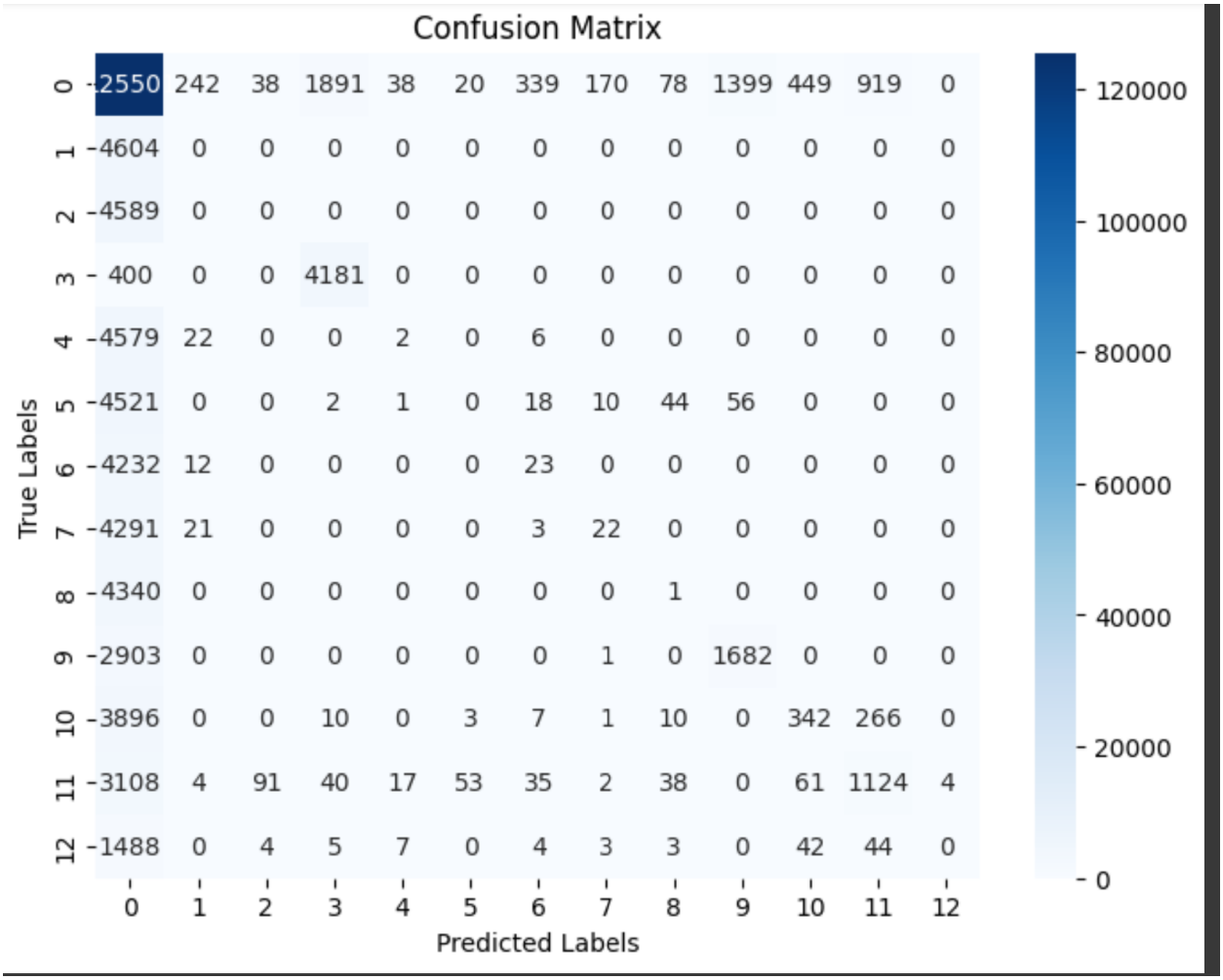
```python
# Evaluate the model on the validation data
val_accuracy = accuracy_score(y_val, y_val_pred)
print(f'Validation Accuracy: {val_accuracy}')


print(classification_report(y_val, y_val_pred))
```

```
Validation Accuracy: 0.7279257740099363
              precision    recall  f1-score   support

           0       0.74      0.96      0.84    130896
           1       0.00      0.00      0.00      4580
           2       0.00      0.00      0.00      4786
           3       0.68      0.92      0.78      4556
           4       0.01      0.00      0.00      4595
           5       0.01      0.00      0.00      4657
           6       0.05      0.01      0.01      4210
           7       0.11      0.01      0.01      4404
           8       0.01      0.00      0.00      4406
           9       0.54      0.37      0.44      4492
          10       0.40      0.08      0.14      4587
          11       0.49      0.23      0.32      4613
          12       0.17      0.00      0.00      1580

    accuracy                           0.73    182362
   macro avg       0.25      0.20      0.19    182362
weighted avg       0.59      0.73      0.64    182362
```


Confusion Matrix
True Labels
Predicted Labels
0 2550 242 38 1891 38 20 339 170 78 1399 449 919 0
1 4604 0 0 0 0 0 0 0 0 0 0 0 0
2 4589 0 0 0 0 0 0 0 0 0 0 0 0
3 400 0 0 4181 0 0 0 0 0 0 0 0 0
4 4579 22 0 0 2 0 6 0 0 0 0 0 0
5 4521 0 0 2 1 0 18 10 44 56 0 0 0
6 4232 12 0 0 0 0 23 0 0 0 0 0 0
7 4291 21 0 0 0 0 3 22 0 0 0 0 0
8 4340 0 0 0 0 0 0 0 1 0 0 0 0
9 2903 0 0 0 0 0 0 1 0 1682 0 0 0
10 3896 0 0 10 0 3 7 1 10 0 342 266 0
11 3108 4 91 40 17 53 35 2 38 0 61 1124 4
12 1488 0 4 5 7 0 4 3 3 0 42 44 0
0 1 2 3 4 5 6 7 8 9 10 11 12
120000
100000
80000
60000
40000
20000
0