

DATABURNERS

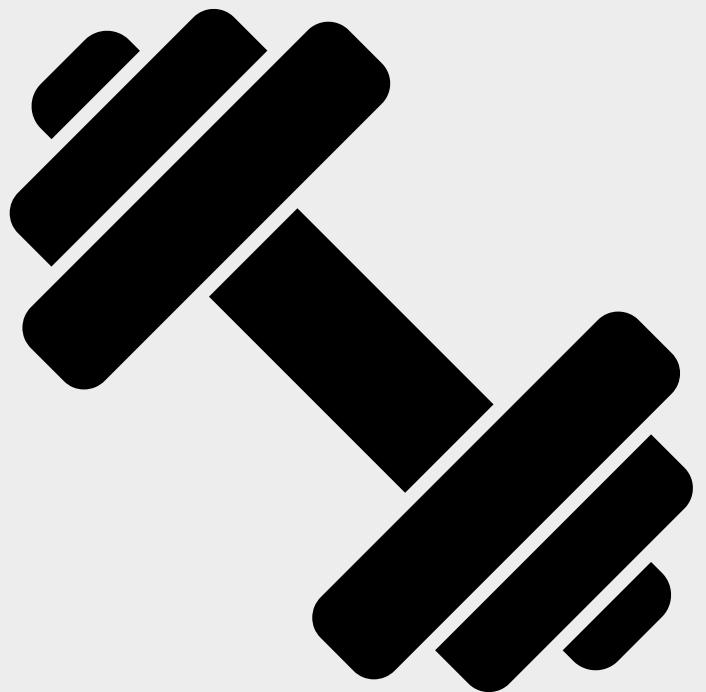


TOPICS

- Problem Definition and Motivation
- Data Handling and Preparation
- Methodology and Implementation
- Evaluation and Analysis
- Q&A



MOTIVATION



ทุกวันนี้คนหันมาใส่ใจสุขภาพมากขึ้น หลายคนอยากรู้ว่าตัวเอง
เพาแพลณ์แคลอรี่จากการออกกำลังกายไปเท่าไหร่ แต่การวัด
แคลอรี่ที่แม่นยำมักต้องใช้อุปกรณ์ราคาแพง หรือสูตรที่ไม่ได้
เหมาะสมกับทุกคน เราเลยอยากสร้างโมเดลง่าย ๆ ที่ใช้ข้อมูลทั่วไป
อย่างอายุ เพศ และอัตราการเต้นหัวใจ เพื่อช่วยคำนวณแคลอรี่ที่
เพาแพลณ์ได้อย่างแม่นยำ โดยไม่ต้องใช้อุปกรณ์พิเศษ

SCOPE

ในโปรเจกต์นี้ เราลองใช้ 2 โมเดลในการคำนายแคลอรี่ที่เพา
พลาญ ได้แก่ Linear Regression และ Neural Network
(MLP) โดยใช้ข้อมูลจากการออกกำลังกายของสมาชิกยิม เช่น
อายุ เพศ เวลาออกกำลังกาย อัตราการเต้นหัวใจ และอุณหภูมิ
ร่างกาย พร้อมเตรียมข้อมูลให้เหมาะสมก่อนฝึกโมเดลด้วย
Keras และ Sklearn



ABOUT DATASET

“Gym Members Exercise Dataset”

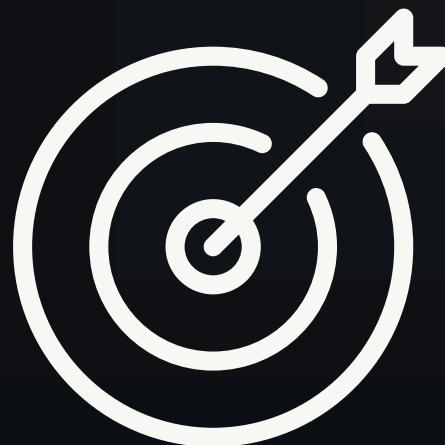
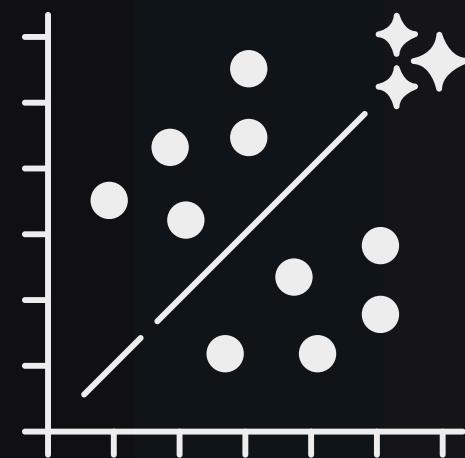
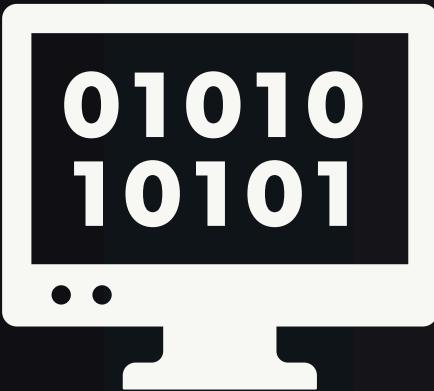
ชุดข้อมูลนี้รวมข้อมูลจากสมาชิกยิมกั้งหนด 973 ตัวอย่าง โดยบันทึกรายละเอียดกั้งด้านร่างกายและพฤติกรรมการออกกำลังกาย เช่น อัตราการเต้นของหัวใจ ระยะเวลาออกกำลังกาย และแคลอรี่ที่เผาผลาญ พร้อมข้อมูลส่วนตัวอย่างอายุ เพศ น้ำหนัก ส่วนสูง และระดับประสบการณ์

Features

- Age & Gender – ข้อมูลพื้นฐานของสมาชิก
- Weight, Height & BMI – ตัวบ่งชี้ทางกายภาพ
- Heart Rate – มีกั้งค่าเฉลี่ย ค่าสูงสุด และขณะพัก
- Workout Info – ประเภท ระยะเวลา ความถี่ต่อสัปดาห์
- Calories Burned – แคลอรี่ที่เผาผลาญในแต่ละเซสชัน
- Fat % & Water Intake – ตัวชี้วัดด้านสุขภาพ
- Experience Level – ระดับตั้งแต่เริ่มต้นจนถึงผู้เชี่ยวชาญ



PREPROCESS



แปลงข้อมูล
หมวดหมู่

จัดการ
OUTLIER

แยก FEATURES
และ TARGET

1

2

3

```
1 data = pd.get_dummies(data, columns=['Gender'])
2 data = data.drop(columns=['Gender_Female', 'Workout_Type'], axis=1)
3 data['Gender_Male'] = data['Gender_Male'].astype(int)
4 data
```

- ใช้ get_dummies() แปลงคอลัมน์ Gender เป็นตัวเลข (One-Hot Encoding)
- ลบคอลัมน์ที่ไม่จำเป็น เช่น Workout_Type และ Gender_Female เพื่อหลีกเลี่ยง multicollinearity
- แปลงค่า Gender_Male ให้เป็นชนิดข้อมูลตัวเลข (int)

```
1 Q1 = data.quantile(0.25)
2 Q3 = data.quantile(0.75)
3
4 IQR = Q3 - Q1
5 outlier = ((data < Q1 - 1.5 * IQR) | (data > Q3 + 1.5 * IQR))
6 data = data[~(outlier).any(axis=1)]
```

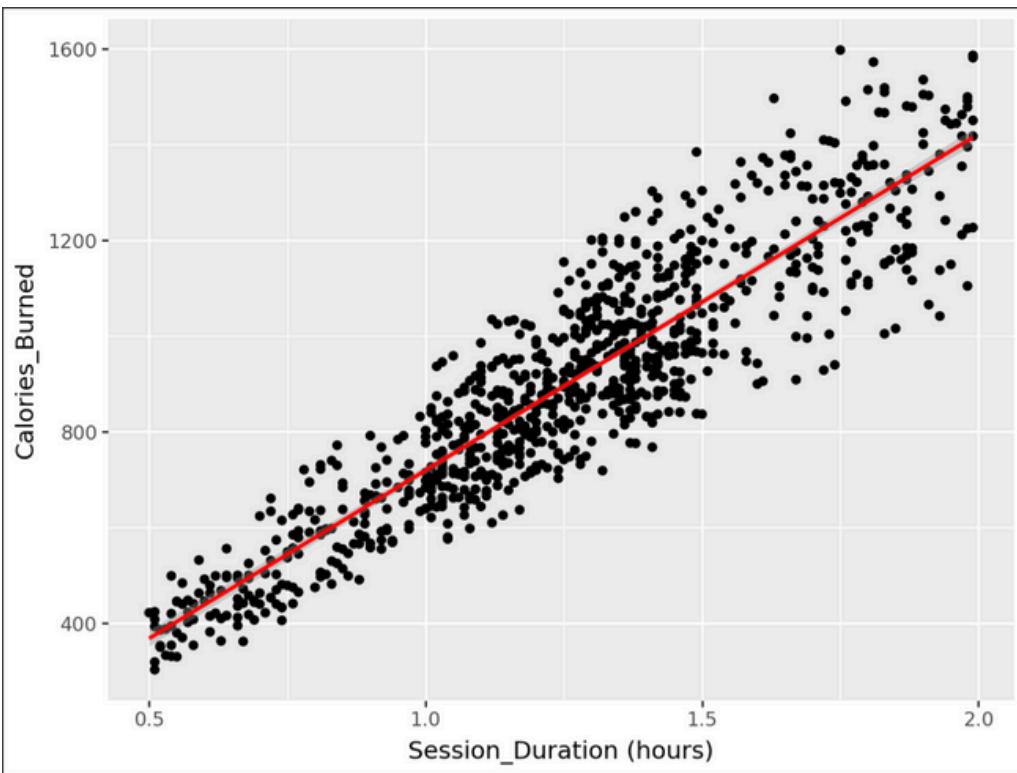
- ใช้เทคนิค IQR (Interquartile Range) ตรวจจับค่าเกินจากช่วงปกติ
- ลบข้อมูลที่เป็น outlier ออกเพื่อให้โมเดลเรียนรู้ได้แม่นยำขึ้น

- แยก Calories_Burned เป็นตัวแปรเป้าหมาย (Target y)
- ส่วนที่เหลือเก็บไว้เป็นตัวแปรอิสระ (Feature X)
- แบ่งข้อมูลออกเป็นชุดฝึก (Train) และทดสอบ (Test) ด้วย train_test_split โดยใช้ 80/20

การเลือกเทคนิค MACHINE LEARNING

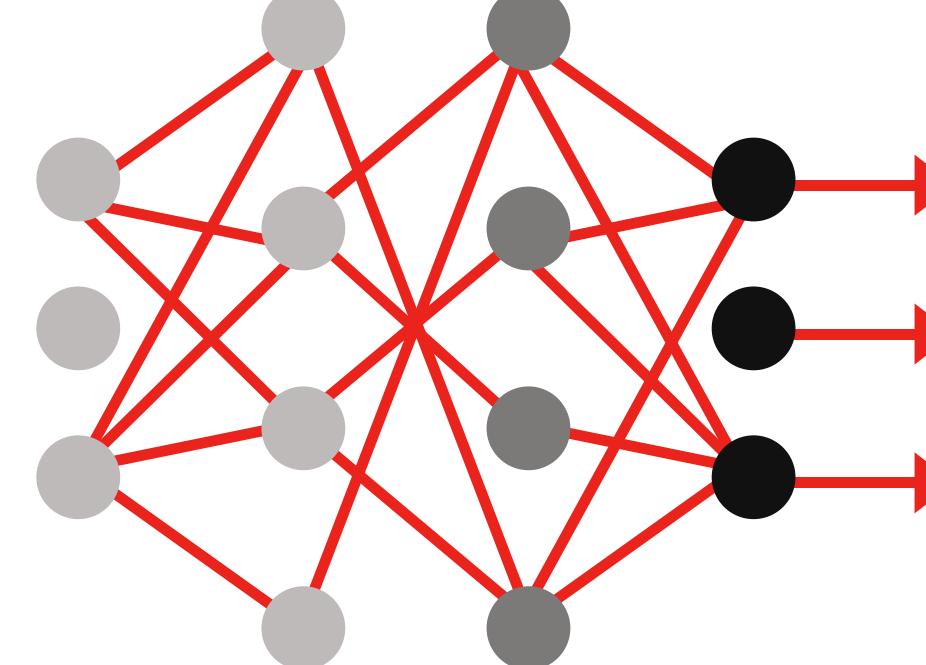
Linear Regression

เราเลือก Linear Regression เพราะเป็นโมเดลพื้นฐานที่สามารถใช้เป็นจุดเริ่มต้นในการประเมินความสัมพันธ์ระหว่างตัวแปรต่าง ๆ กับแคลอรี่ที่เผาผลาญได้อย่างชัดเจน และช่วยให้เราเห็นภาพรวมของข้อมูลก่อนใช้โมเดลที่ซับซ้อนขึ้น



Neural Network (MLP)

เนื่องจากพฤติกรรมการเผาผลาญแคลอรี่อาจมีความสัมพันธ์แบบไม่เป็นเส้นตรง เช่น อัตราการเต้นของหัวใจหรือเบอร์เซ็นต์ไขมัน เราจึงเลือกใช้ Neural Network เพื่อให้สามารถจับความซับซ้อนของข้อมูลได้ดียิ่งขึ้น โดยเฉพาะเมื่อข้อมูลมีหลาย feature ที่อาจส่งผลร่วมกัน



Linear Regression

```
results = []

for name, model in models.items():
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    cv_mse_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
    average_cv_mse = -cv_mse_scores.mean()

    cv_r2_scores = cross_val_score(model, X, y, cv=5, scoring='r2')
    average_cv_r2 = cv_r2_scores.mean()

    results.append({
        'Model': name,
        'Test MSE': mse,
        'Test R2': r2,
        'Average CV MSE': average_cv_mse,
        'Average CV R2': average_cv_r2
    })

print(f"\n{name} Results:")
print(f'Mean Squared Error on Test Set: {mse}')
print(f'R-squared (R2) on Test Set: {r2}')
print(f'Average Mean Squared Error from Cross-Validation: {average_cv_mse}')
print(f'Average R2 from Cross-Validation: {average_cv_r2}')

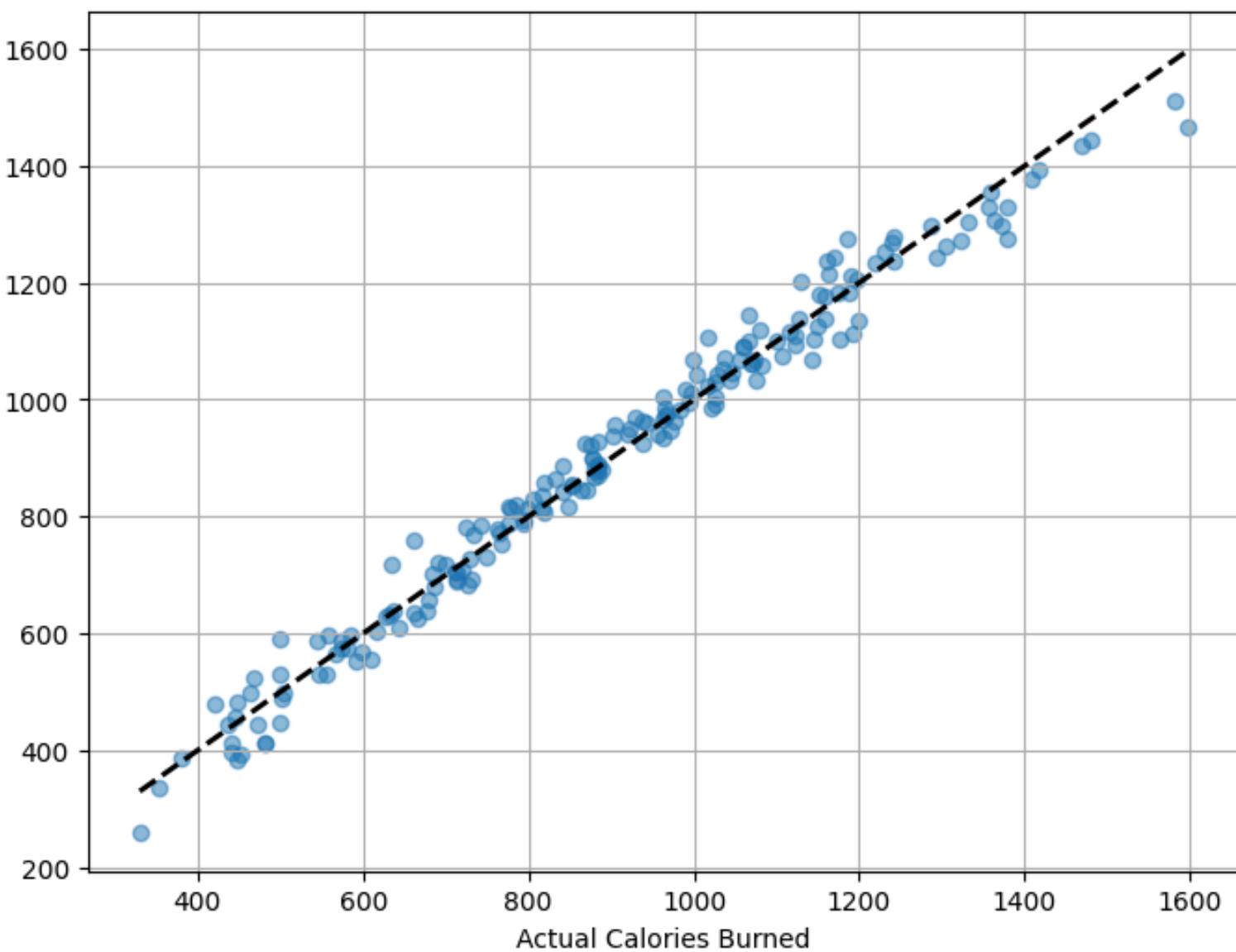
results_df = pd.DataFrame(results)
display(results_df)
```

▼ LinearRegression [i](#) [?](#)

LinearRegression()

'Linear Regression'

Actual vs Predicted Calories Burned



	True Values	Predicted Values	Diff		
Model	Test MSE	Test R2	Average CV MSE	Average CV R2	
0	864	1242.0	1278.501688	36.501688	
1	74	777.0	816.952399	39.952399	
2	659	904.0	955.797466	51.797466	
3	528	1107.0	1074.130783	32.869217	
4	736	1598.0	1466.107738	131.892262	
5
6	2	677.0	639.282235	37.717765	
7	871	1025.0	992.989517	32.010483	
8	769	1082.0	1057.968610	24.031390	
9	566	1071.0	1061.646107	9.353893	
10	5	1116.0	1117.652526	1.652526	

MODEL 8 HIDDEN LAYER LEARNING_RATE=0.001

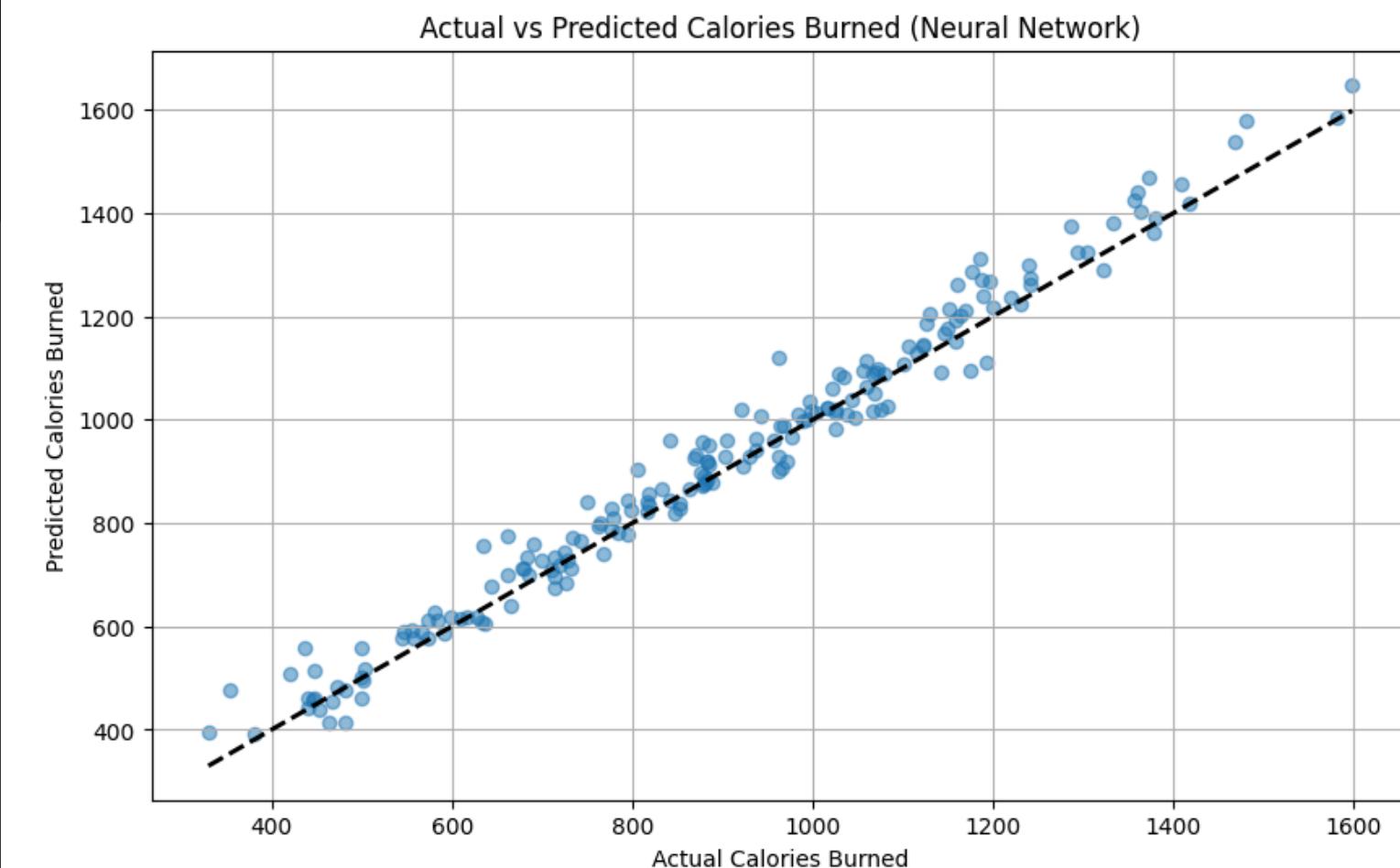
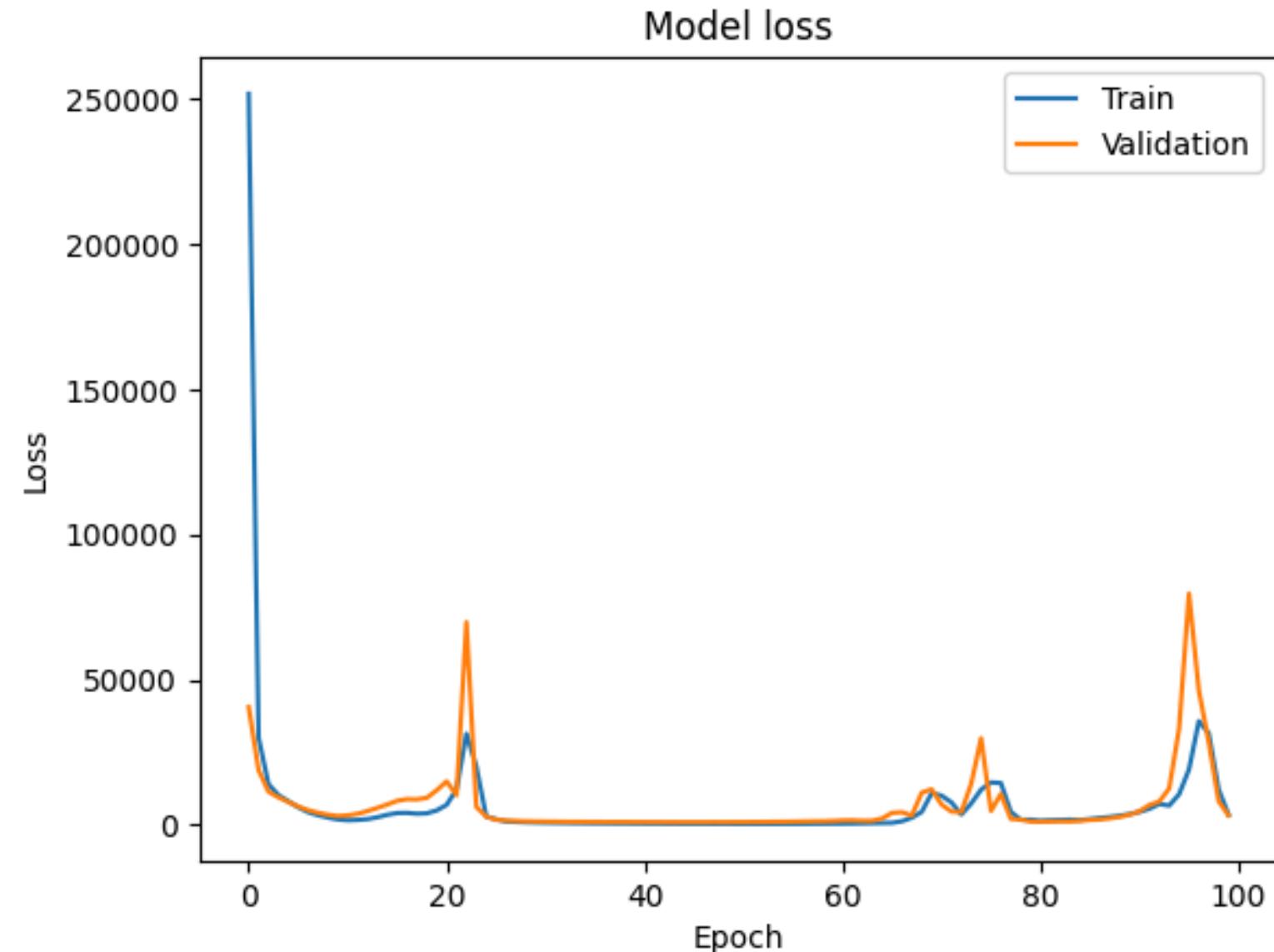
```
final_model = Sequential([
    Dense(8192, activation='relu', input_shape=(X_train_fold_scaled.shape[1],)),
    Dense(4096, activation='relu'),
    Dense(2048, activation='relu'),
    Dense(1024, activation='relu'),
    Dense(512, activation='relu'),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])
final_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
final_model.compile(optimizer=final_optimizer, loss='mean_squared_error', metrics=['mean_squared_error'])
```

100 EPOCHS

Average Mean Squared Error (K-Fold): 5085.580981445312

```
y_pred_nn = final_model.predict(X_test_scaled)
for i in range(10):
    print(y_test[i], y_pred_nn[i], "diff : ", abs(y_test[i] - y_pred_nn[i]))
```

6/6 ————— 1s 59ms/step
1242.0 [1261.6855] diff : [19.68554688]
777.0 [829.55414] diff : [52.55413818]
904.0 [961.24365] diff : [57.24365234]
1107.0 [1142.2548] diff : [35.25476074]
1598.0 [1648.4314] diff : [50.43139648]
420.0 [507.55524] diff : [87.55523682]
1379.0 [1391.3771] diff : [12.3770752]
1231.0 [1222.6821] diff : [8.31787109]
592.0 [585.56256] diff : [6.43743896]
1175.0 [1094.7533] diff : [80.2467041]



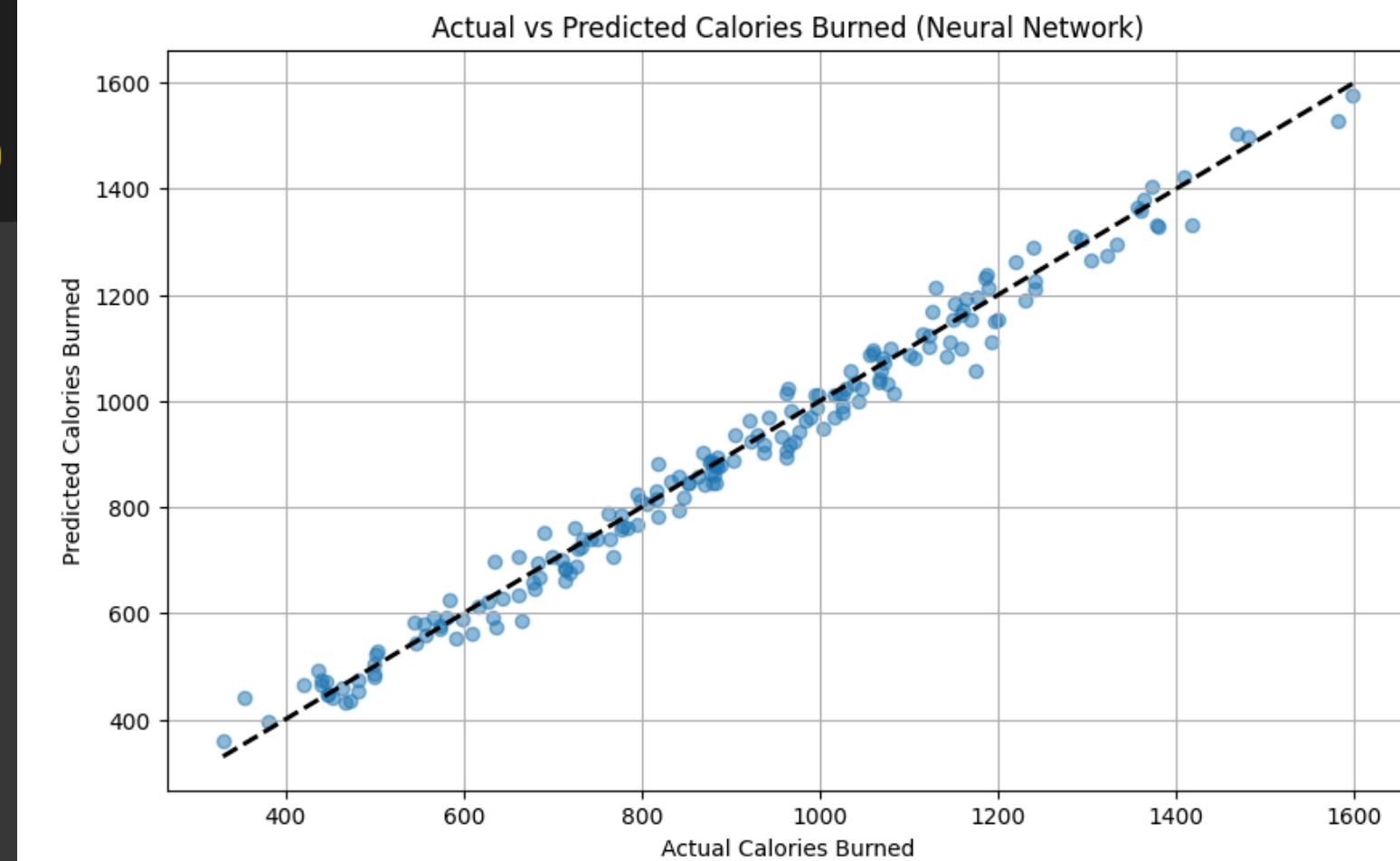
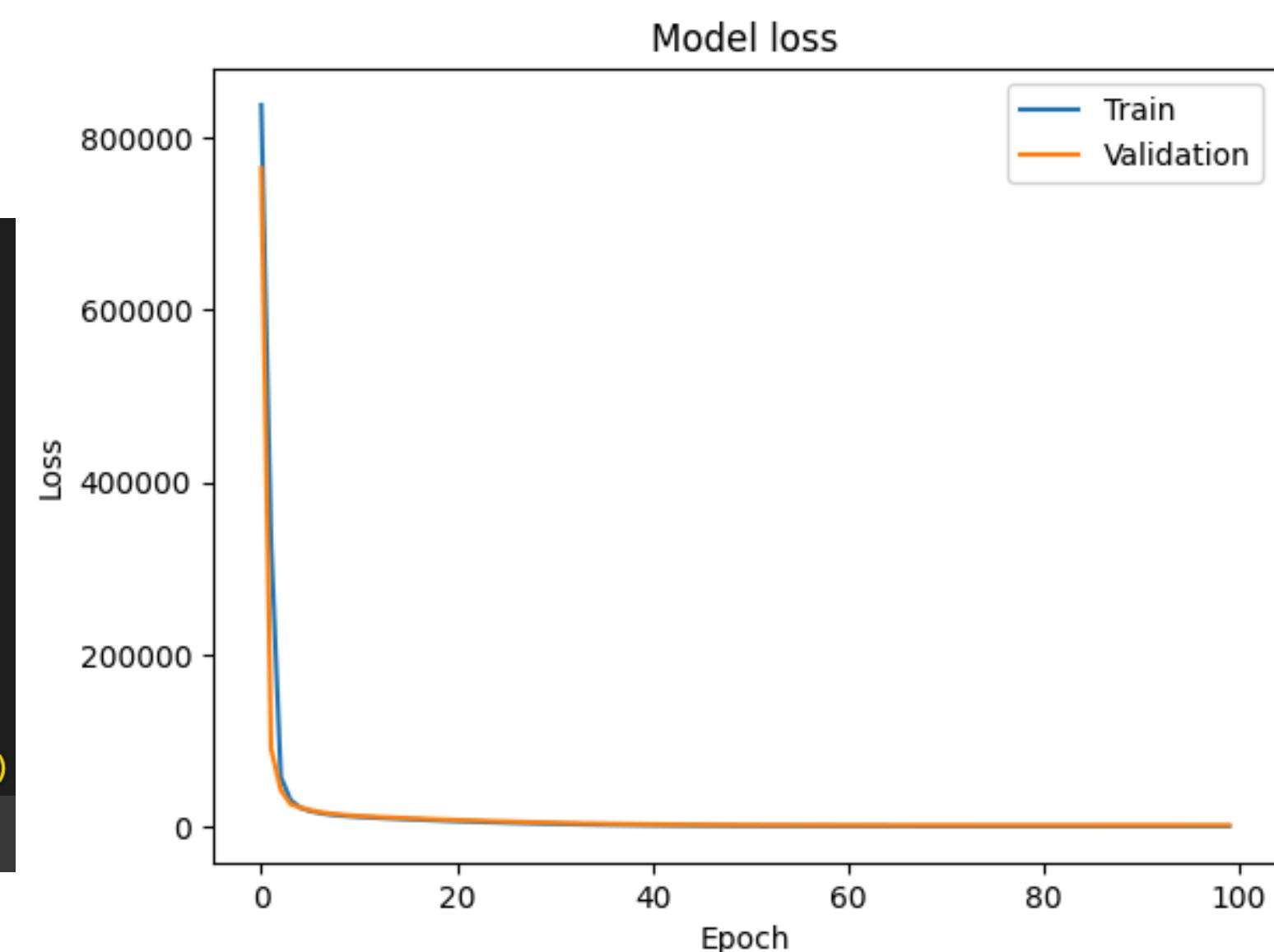
MODEL 8 HIDDEN LAYER LEARNING_RATE=0.0001

```
model_fold = Sequential([
    Dense(8192, activation='relu', input_shape=(X_train_fold_scaled.shape[1],)),
    Dense(4096, activation='relu'),
    Dense(2048, activation='relu'),
    Dense(1024, activation='relu'),
    Dense(512, activation='relu'),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])
optimizer_fold = tf.keras.optimizers.Adam(learning_rate=0.00001)
model_fold.compile(optimizer=optimizer_fold, loss='mean_squared_error', metrics=['mean_squared_error'])
```

Average Mean Squared Error (K-Fold): 1031.4171264648437

```
y_pred_nn = final_model.predict(X_test_scaled)
for i in range(10):
    print(y_test[i], y_pred_nn[i], "diff : ", abs(y_test[i] - y_pred_nn[i]))
```

6/6 ————— 1s 53ms/step
1242.0 [1211.1405] diff : [30.85949707]
777.0 [784.9318] diff : [7.93182373]
904.0 [935.8901] diff : [31.89007568]
1107.0 [1082.3107] diff : [24.68933105]
1598.0 [1575.4563] diff : [22.54370117]
420.0 [465.4556] diff : [45.45559692]
1379.0 [1327.4899] diff : [51.51013184]
1231.0 [1189.4607] diff : [41.53930664]
592.0 [551.6787] diff : [40.32128906]
1175.0 [1058.1201] diff : [116.87988281]



MODEL

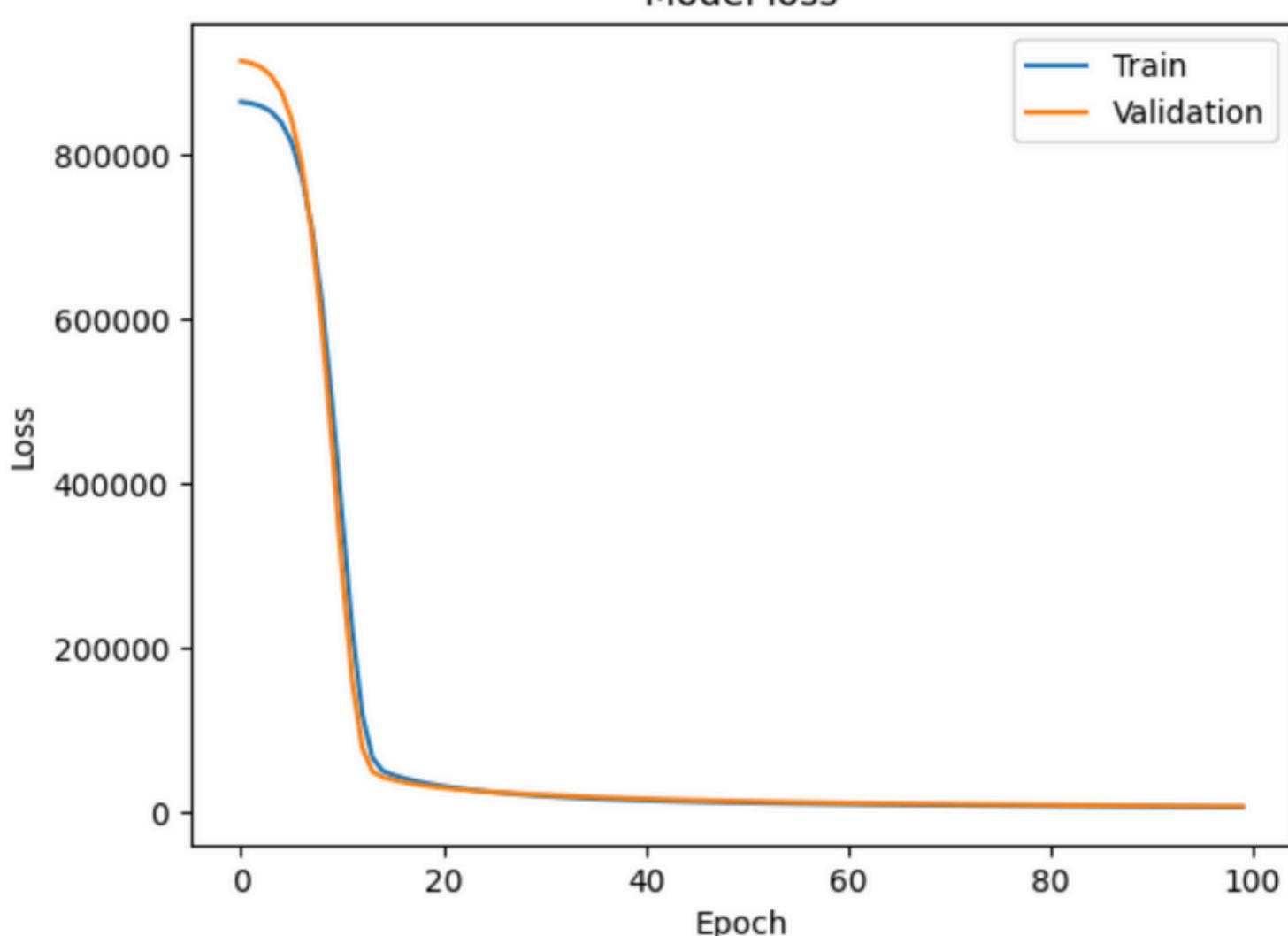
3 HIDDEN LAYER LEARNING_RATE=0.0001

```

final_model = Sequential([
    Dense(512, activation='relu', input_shape=(X_train_fold_scaled.shape[1],)),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(1)
])
final_optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
final_model.compile(optimizer=final_optimizer, loss='mean_squared_error', metrics=['mean_squared_error'])
24/24 ————— 0s 4ms/step - loss: 340.2118 - mean_squared_error: 340.2118
Average Mean Squared Error (K-Fold): 1135.3297485351563

```

100 EPOCHS



```

1 y_pred_nn = final_model.predict(X_test_scaled)
2 for i in range(10):
3     print(y_test[i], y_pred_nn[i], "diff : ", abs(y_test[i] - y_pred_nn[i]))

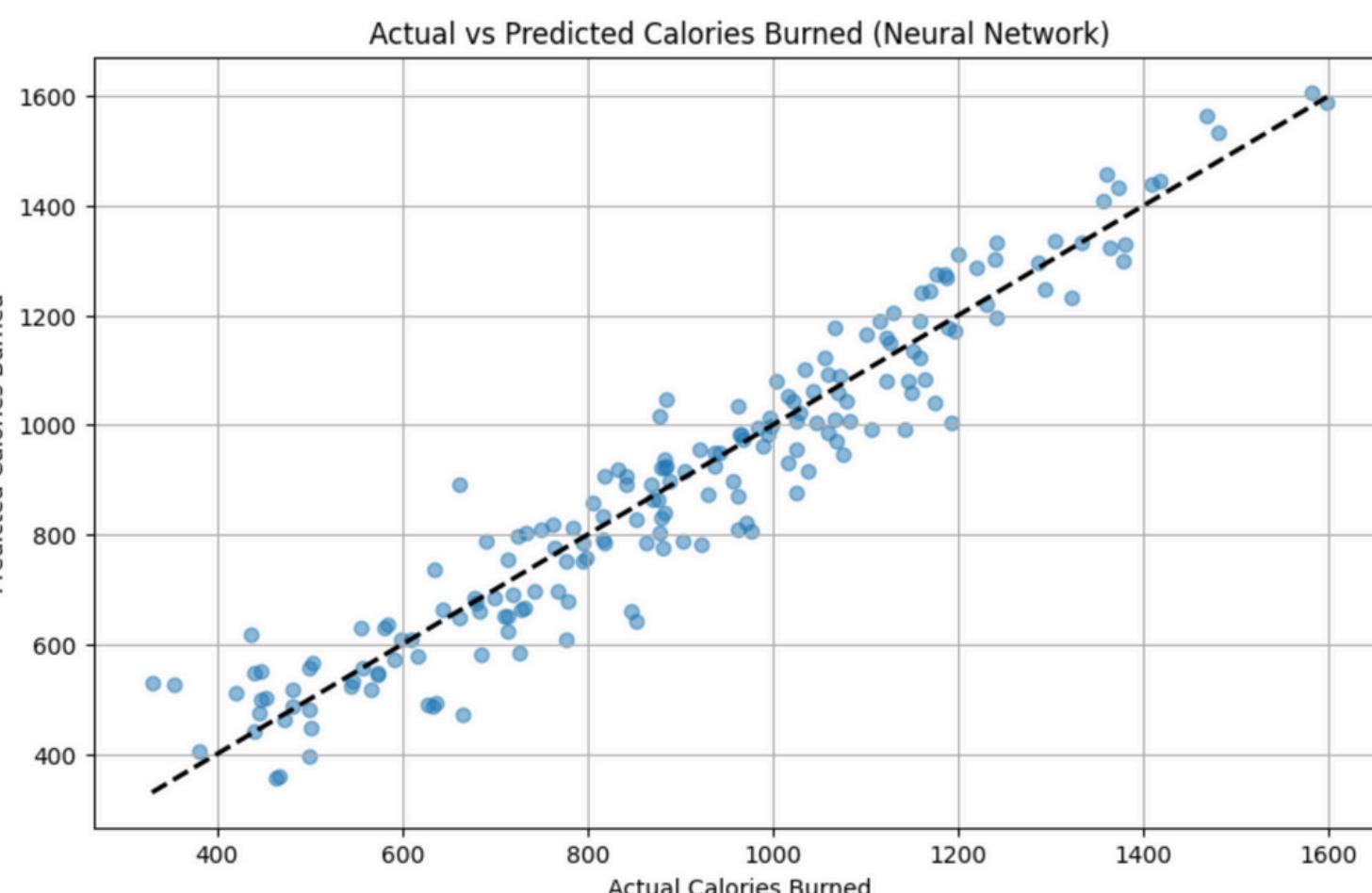
```



```

6/6 ————— 0s 48ms/step
1242.0 [1333.6465] diff : [91.64648438]
777.0 [752.74445] diff : [24.2555542]
904.0 [916.2185] diff : [12.21850586]
1107.0 [991.2943] diff : [115.70568848]
1598.0 [1587.659] diff : [10.34094238]
420.0 [511.18417] diff : [91.18417358]
1379.0 [1328.3988] diff : [50.60119629]
1231.0 [1219.2549] diff : [11.74511719]
592.0 [571.8321] diff : [20.16790771]
1175.0 [1039.3912] diff : [135.60876465]

```



MODEL

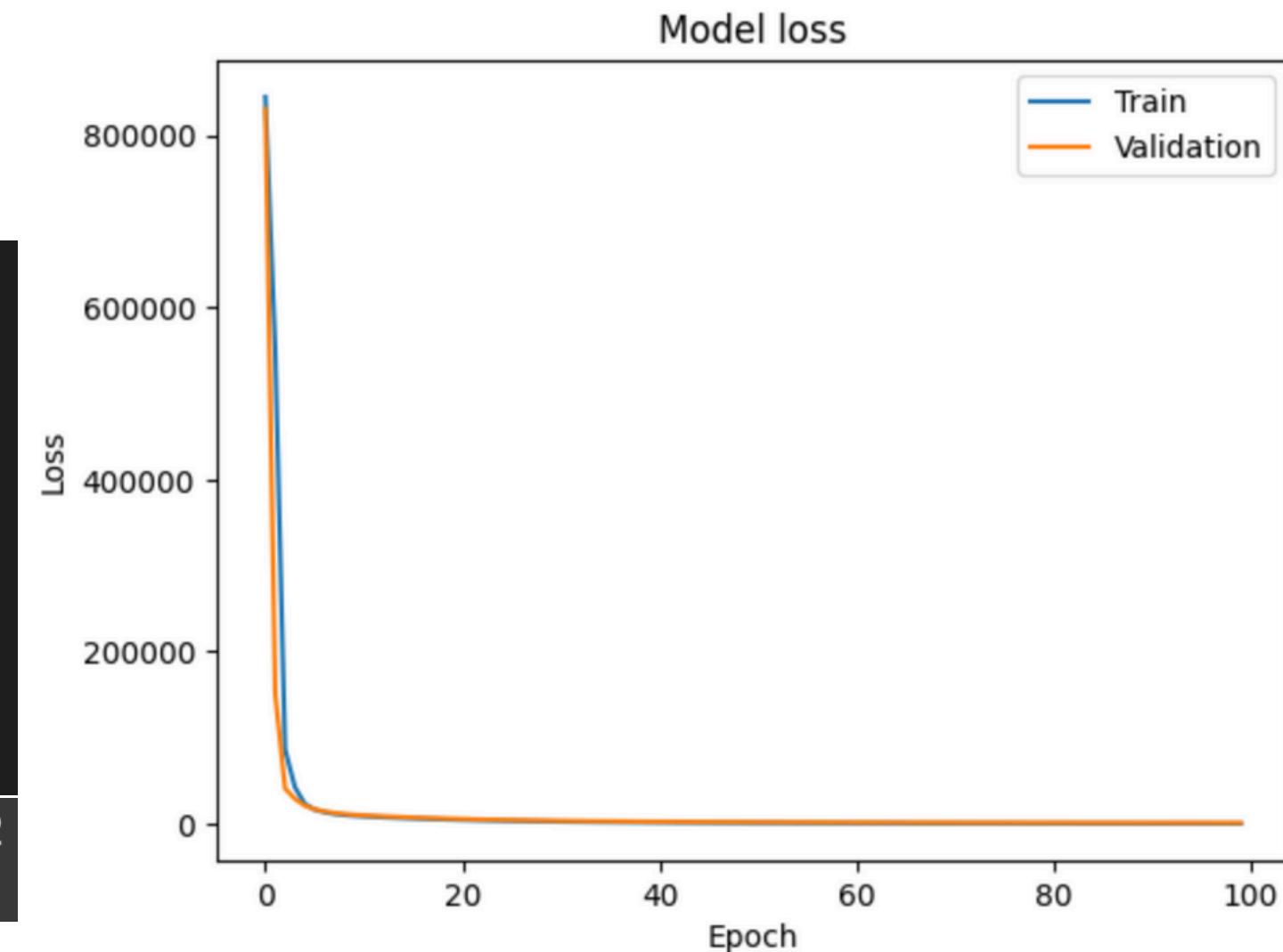
3 HIDDEN LAYER LEARNING_RATE=0.001

The Best overall performance model

```
1 final_model = Sequential([
2     Dense(512, activation='relu', input_shape=(X_train_fold_scaled.shape[1],)),
3
4     Dense(256, activation='relu'),
5
6     Dense(128, activation='relu'),
7
8     Dense(64, activation='relu'),
9     Dense(1)
10])
11 final_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
12 final_model.compile(optimizer=final_optimizer, loss='mean_squared_error', metrics=['mean_squared_error'])
13
```

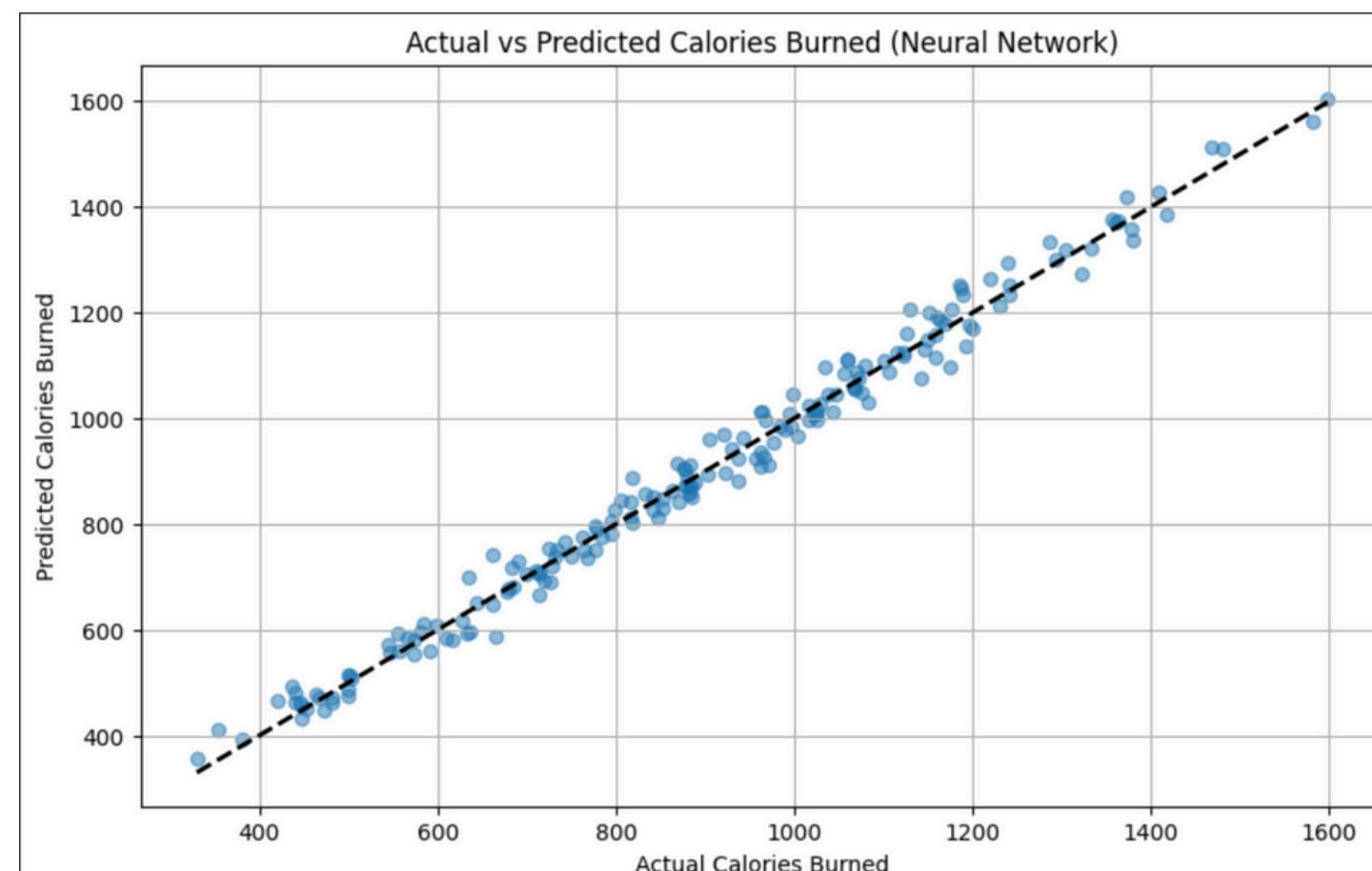
24/24 ————— 0s 4ms/step - loss: 224.2202 - mean_squared_error: 224.2202
Average Mean Squared Error (K-Fold): 1001.1873046875

100 EPOCHS

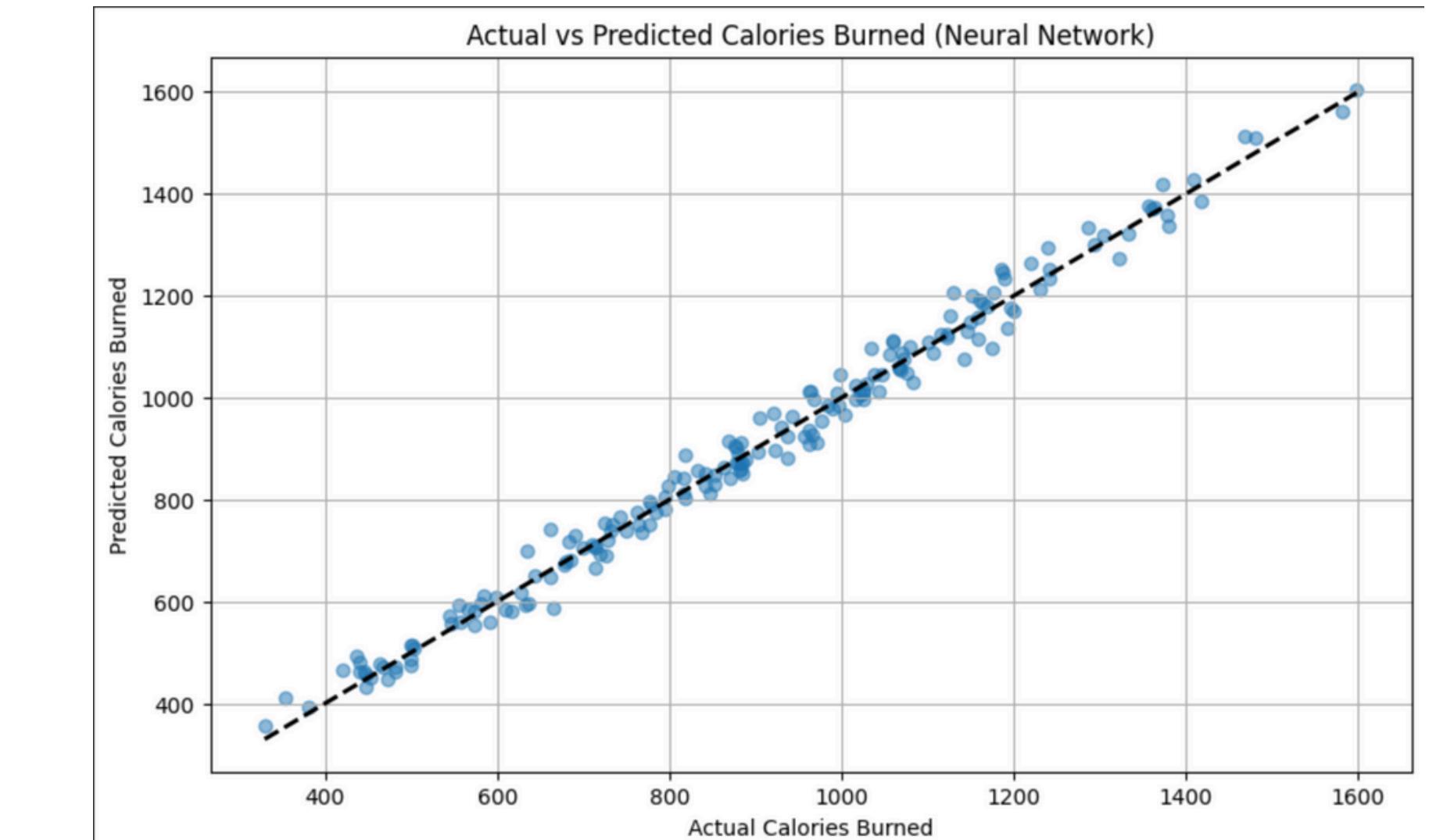
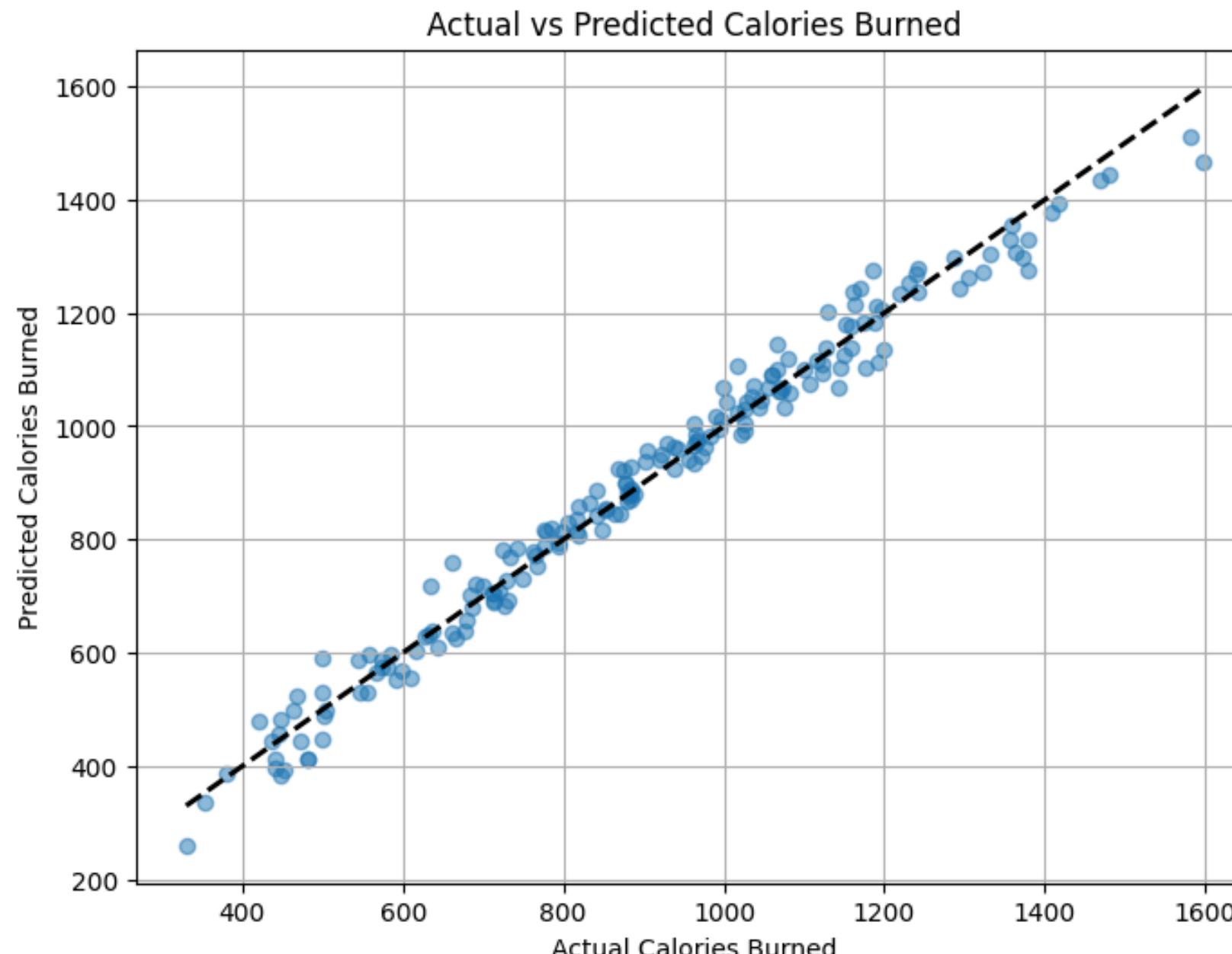


6/6 ————— 1s 62ms/step

1242.0 [1251.959] diff : [9.95898438]
777.0 [796.401] diff : [19.40100098]
904.0 [961.6618] diff : [57.6618042]
1107.0 [1087.8983] diff : [19.10168457]
1598.0 [1604.8148] diff : [6.81481934]
420.0 [466.10217] diff : [46.10217285]
1379.0 [1337.6516] diff : [41.34838867]
1231.0 [1211.4873] diff : [19.51269531]
592.0 [560.04956] diff : [31.95043945]
1175.0 [1096.034] diff : [78.96594238]



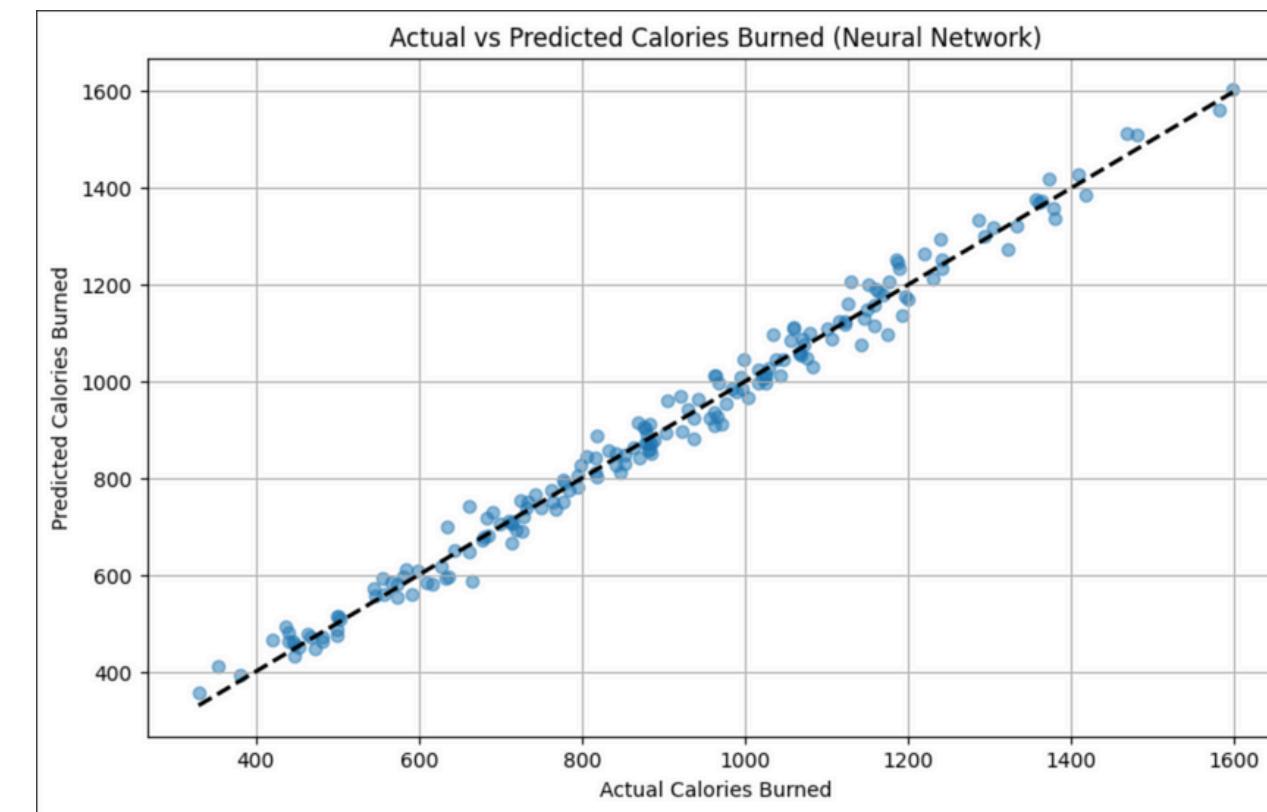
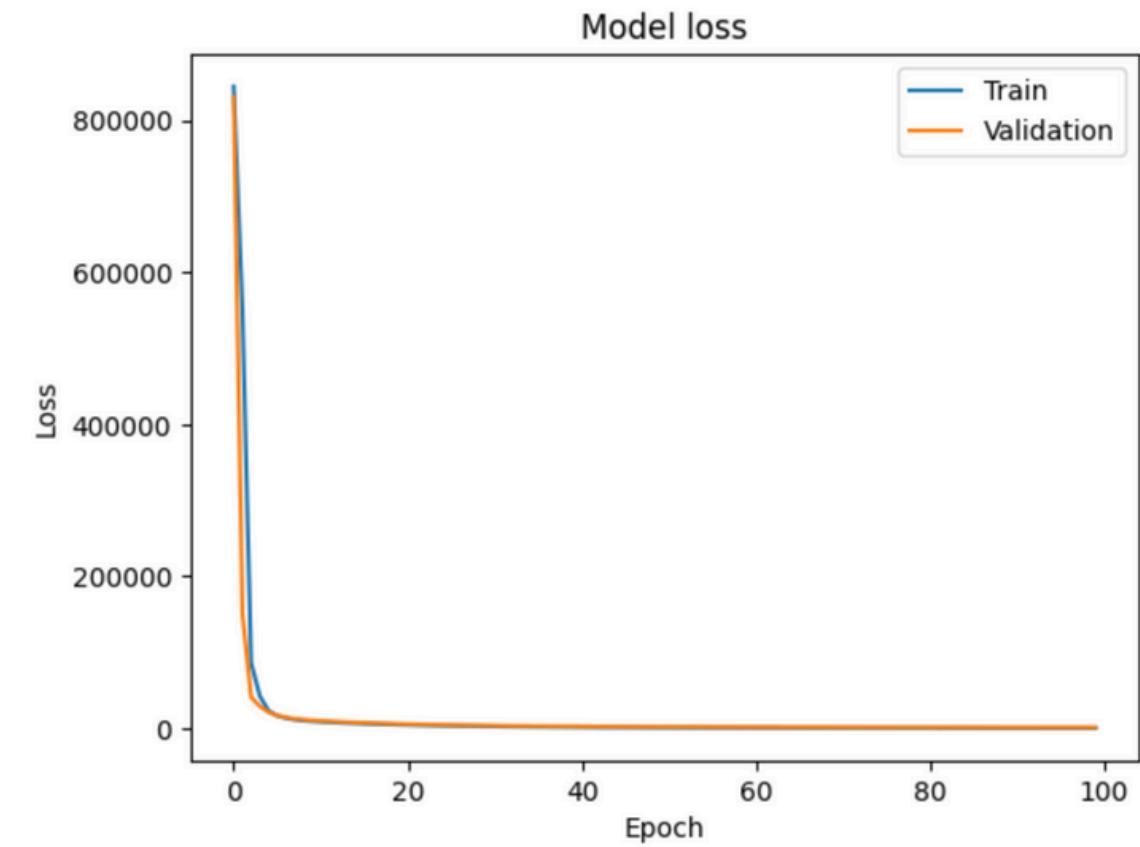
Linear Regression vs Neural Network (MLP)



24/24 ————— 0s 4ms/step - loss: 224.2202 - mean_squared_error:
Average Mean Squared Error (K-Fold): 1001.1873046875

Model	Test MSE	Test R2	Average CV MSE	Average CV R2
0 Linear Regression	1466.266868	0.980481	1438.379603	0.979180

CONCLUSION



ประสิทธิภาพโมเดล Neural Network ยอดเยี่ยม ($R^2 = 0.985$, $MSE = 1001.187$)

อธิบายความแปรปรวนของข้อมูลได้ถึง 98.5% (ดีมาก)

แม่นยำกว่า Linear Regression ($R^2=0.912$, $MSE=3021.449$) อย่างชัดเจน เนื่องจากสามารถเรียนรู้ ความสัมพันธ์ที่ไม่เป็นเชิงเส้นได้

พฤติกรรมการเรียนรู้และการทำนาย

Loss Curve : Train Loss & Validation Loss ลดลงอย่างรวดเร็วและเข้าสู่สภาวะคงที่ ใกล้เคียงกับปั่งชี้การเรียนรู้ที่ มีเสถียรภาพ และ ไม่พบสัญญาณ Overfitting ที่ชัดเจน (แนวทางแก้ไข Overfitting ในอนาคต: Dropout, Early Stopping, Data Augmentation, Regularization)

Actual vs. Predicted : จุดข้อมูลส่วนใหญ่ เกาะกลุ่มใกล้เส้นกางแยก (y=x) แสดงความแม่นยำสูงการกระจาย Error เป็นแบบสุ่ม ไม่มี bias ชัดเจนมีบางจุดเบี่ยงเบนบ้าง แต่โดยรวมยัง บ่าเชื่อถือ

THANK YOU !!!

DATABURNERS

Professional Bodybuilder

