Fuel-Economy Prediction With Linear Regression and Neural Network

Avinash Bunga

Master of Science in Information Systems and Business Analytics

Park University

CIS625HOS2P2025 Machine Learning for Business

Professor: Abdelmonaem Jornaz

April 27, 2025

Fuel‑Economy Prediction With Linear Regression and Neural Network

This study demonstrates when a neural network justifies its added complexity. The *auto-MPG* data set (UCI Machine Learning Repository, 1980) was analyzed to compare a multiple linear regression baseline with a three-hidden-layer neural network. Five available engine or age-related variables were used to predict a car's miles‑per‑gallon (mpg): horsepower, weight, displacement, cylinders, and model year. The auto-MPG set was chosen because it is small, public, and contains clear non-linear links between engine specs and fuel economy (See Appendix A for Python script).

Data Preparation: After removing rows lacking horsepower, 392 cars remained. The dataset was partitioned into 70% training (n = 274) and 30% test (n = 118) subsets. All predictors were scaled to the 0–1 interval with min-max normalization so that features share a standard numeric range (See Appendix A for Python script; Waskom, 2021).

Modelling Strategy: Two supervised models were evaluated:

1. **Multiple linear regression** implemented with scikit‑learn's LinearRegression class (GeeksforGeeks, 2022).
2. **Deep neural network** consisting of three hidden layers (128 → 64 → 32 → 16 neurons, ReLU activations, Adam optimiser). Training used an early-stopping callback (patience=15) to halt when the validation loss stopped improving (ScienceDirect, n.d.).

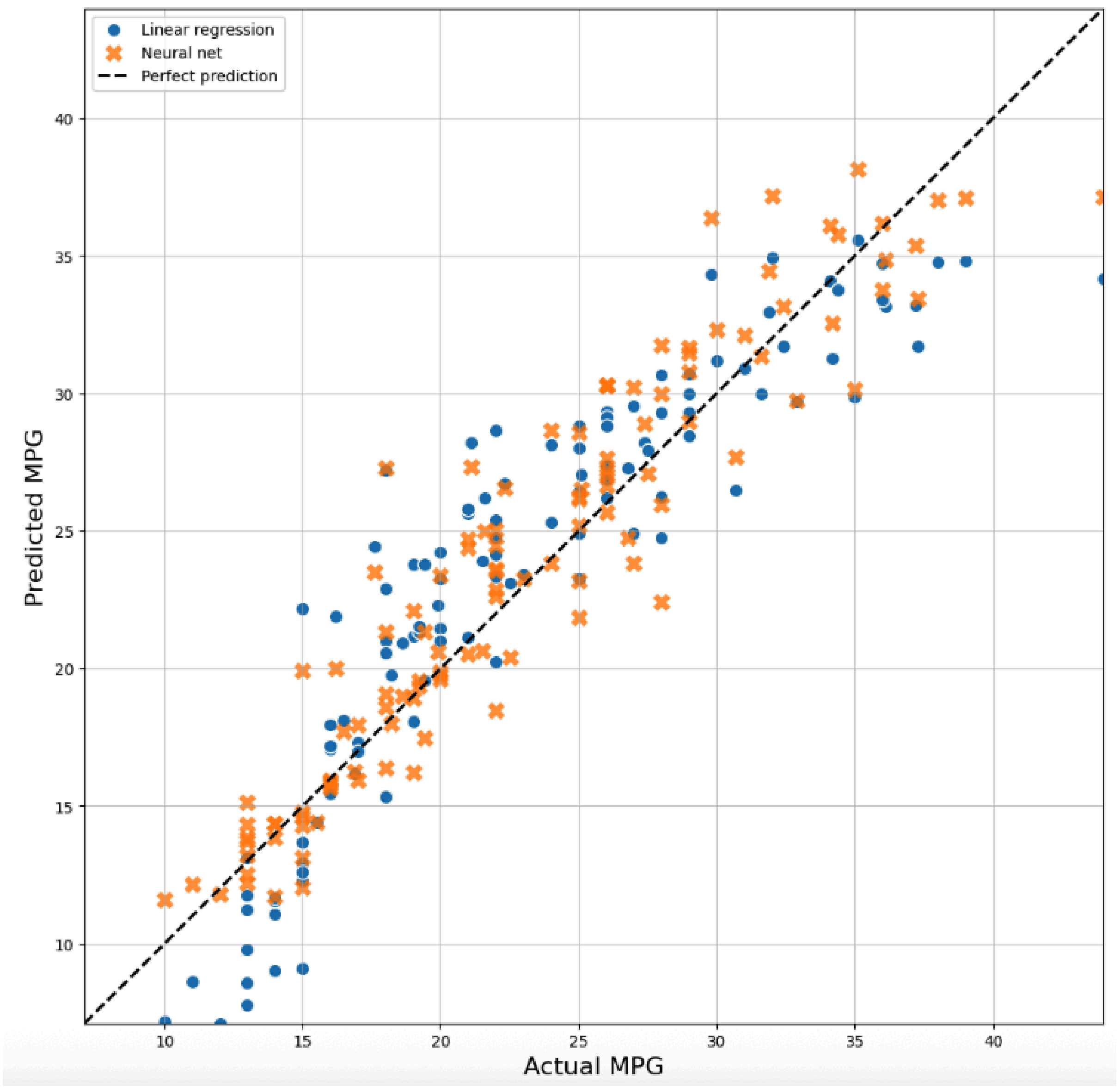
Table 1

Root‑mean‑square error (RMSE) for the two models

Model	Test MSE	Test RMSE (mpg)
Linear regression (5 features)	10.23	3.2
Deep neural network (128 → 64 → 32 → 16)	6.9	2.63

Figure 1

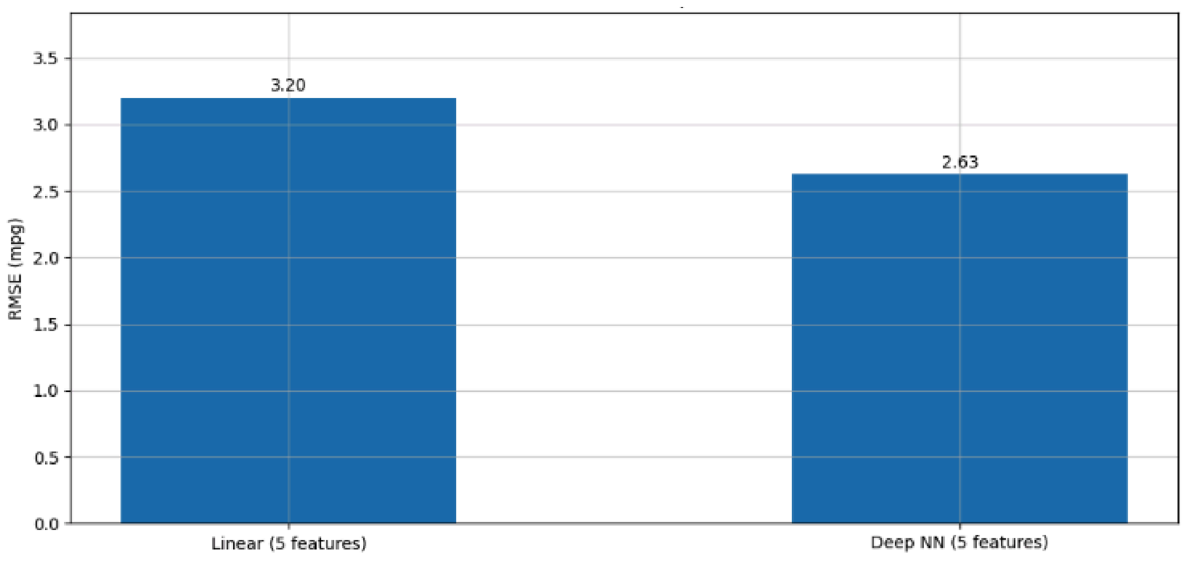
Actual vs. Predicted MPG — Linear vs. Deep NN



Orange × marks (neural net) align more closely to the 45-degree reference line than the blue dots of the linear model (See Appendix B for Python script).

Figure 2

Test‑set error (RMSE) for the two models.



The bar representing the neural network (2.63 mpg) is visibly shorter than that of linear regression (3.20 mpg), visually confirming the 18 % improvement observed in Table 1(See Appendix C for Python script).

Model year introduces a curved upward trend in fuel economy, and this effect interacts with horsepower and weight. A single hyperplane cannot capture these bends, whereas a modest neural network can yield a measurable improvement. Early stopping constrained over‑fitting despite the network's greater capacity (Hussein & Shareef, 2024).

Conclusion

On the test cars, linear regression differed from the true mileage by 3.2 mpg, while the neural network differed by 2.6 mpg. The 18 percent drop in error means the network captured the curved link between engine features, model year, and fuel economy that the straight-line model could not. For this assignment, the neural network is therefore the better practical choice.

References

GeeksforGeeks. (2022, July 11). *Multiple linear regression with scikit-learn*. Retrieved April 26, 2025, from https://www.geeksforgeeks.org/multiple-linear-regression-with-scikit-learn/

Hussein, B. M., & Shareef, S. M. (2024). An empirical study on the correlation between early stopping patience and epochs in deep learning. *ITM Web of Conferences, 64*, 01003. https://doi.org/10.1051/itmconf/20246401003

ScienceDirect. (n.d.). *Three-layered neural network – an overview*. Retrieved April 26, 2025, from https://www.sciencedirect.com/topics/engineering/three-layered-neural-network

Waskom, M. L. (2021). *mpg.csv* [Data set]. https://raw.githubusercontent.com/mwaskom/seaborn-data/master/mpg.cs

Appendix A

```
#Import
import pandas as pd, numpy as np, seaborn as sns, matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.callbacks import EarlyStopping
print("TensorFlow:", tf.__version__)
#Load & clean Seaborn mpg dataset
cars = sns.load_dataset("mpg")
cars = cars.dropna(subset=["horsepower"])
print("Clean shape:", cars.shape)
#Feature matrix (5 predictors) & target
X = cars[["horsepower", "weight",
        "displacement", "cylinders",
        "model_year"]].values
y = cars["mpg"].values
#Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42
)
print("train:", X_train.shape, "| test:", X_test.shape)
#Min-Max scale features
scaler = MinMaxScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s  = scaler.transform(X_test)

#Baseline Linear Regression
lin = LinearRegression().fit(X_train_s, y_train)

lin_mse  = mean_squared_error(y_test, lin.predict(X_test_s))
lin_rmse = np.sqrt(lin_mse)

print(f"\nLinear-regression Test MSE : {lin_mse :.3f}")
print(f"Linear-regression Test RMSE: {lin_rmse:.3f}")

#Deep Neural Network (128→64→32→16)
deep = Sequential([
    Input(shape=(5,)),
    Dense(128, activation='relu'),
    Dense(64,  activation='relu'),
    Dense(32,  activation='relu'),
    Dense(16,  activation='relu'),
    Dense(1)
])
deep.compile(optimizer='adam', loss='mse')
stop = EarlyStopping(monitor='val_loss',
                     patience=15,
                     restore_best_weights=True)
deep.fit(X_train_s, y_train,
         validation_split=0.2,
         epochs=400,
         callbacks=[stop],
         verbose=0)
nn_mse  = mean_squared_error(y_test,
                             deep.predict(X_test_s).flatten())
nn_rmse = np.sqrt(nn_mse)

print(f"\nDeep-NN Test MSE  : {nn_mse :.3f}")
print(f"Deep-NN Test RMSE : {nn_rmse:.3f}")
```

Appendix B

```
#Figure 1: Actual vs. Predicted MPG
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize=(10, 14))
#draw blue dots (linear) and orange Xs (NN)
plt.scatter(y_test, lin.predict(X_test_s),
            color="#1f77b4",
            edgecolor="white",
            alpha=1,
            s=80,
            label="Linear regression")
plt.scatter(y_test, deep.predict(X_test_s).flatten(),
            color="#ff7f0e",
            marker="x",
            s=70,
            linewidth=4,
            alpha=0.8,
            label="Neural net")
#45-degree reference line
ax_min = min(np.min(y_test), np.min(lin.predict(X_test_s)))
ax_max = max(np.max(y_test), np.max(lin.predict(X_test_s)))
plt.plot([ax_min, ax_max], [ax_min, ax_max],
         linestyle="--",
         color="black",
         linewidth=2,
         label="Perfect prediction")
plt.title("Actual vs. Predicted MPG — Linear vs. Deep NN", fontsize=14)
plt.xlabel("Actual MPG", fontsize=16)
plt.ylabel("Predicted MPG", fontsize=16)
plt.xlim(ax_min, ax_max)
plt.ylim(ax_min, ax_max)
plt.gca().set_aspect("equal", adjustable="box")
plt.legend()
plt.grid(alpha=0.6)
plt.tight_layout()
plt.show()
```

Appendix C

```
#Figure 2: bar chart of RMSEs
import pandas as pd
import matplotlib.pyplot as plt
rmse_table = pd.DataFrame({
    "Model": ["Linear (5 features)",
              "Deep NN (5 features)"],
    "RMSE":  [lin_rmse, nn_rmse]
})
print(rmse_table.to_string(index=False))
plt.figure(figsize=(10,5))
plt.bar(rmse_table["Model"], rmse_table["RMSE"], width=0.5)
plt.ylabel("RMSE (mpg)")
plt.title("Test-set Error Comparison")
plt.ylim(0, rmse_table["RMSE"].max() * 1.2)
for idx, val in enumerate(rmse_table["RMSE"]):
    plt.text(idx, val + 0.05, f"{val:.2f}", ha="center")
plt.grid(alpha=0.6)
plt.tight_layout()
plt.show()
```