Name: Marvin Nguyen

Table of Content

1. Executive Summary

Business Issue and Goal The goal of RetailPro, a well-known US-based retail chain that specialises in personal gadgets, home appliances, and consumer electronics, is to improve its sales forecasting skills. Understanding the fundamental elements that affect sales is essential for strategic decision-making in businesses that operate through both physical and online channels. Using a dataset of 2,000 records and 20 variables that reflect internal operations and external market impacts, the project's goal is to develop and assess predictive models. Finding the best AI model to forecast total sales with high accuracy and dependability for possible real-world application is the main goal.

Techniques and Trials Five Multi-Layer Perceptron (MLP) neural networks with different depths, activation functions, and regularisation techniques were developed as part of the modelling process, along with a baseline linear regression model. Standard regression measures, such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and the correlation between expected and actual sales numbers, were used to evaluate each model's performance after it was trained and verified using a 70/30 data split. In order to evaluate trade-offs between accuracy, training stability, and computational cost, the experimental design made it possible to compare simple and sophisticated model designs.

With correlation scores ranging from 0.8600 to 0.9986, the neural network models showed a broad range of prediction skills. Model 3—Deep MLP—performed the best among them, obtaining the lowest MAE (179.52) and MSE (62,417.62), as well

as a high correlation of 0.9974. This suggests high generalisation to unknown data in addition to accurate predictions. Despite the linear regression model's numerically lowest MSE and seemingly flawless correlation of 1.0, this result sparked worries about possible data leakage or overfitting because the model was unable to identify the non-linear patterns present in complex retail data. Other MLP models, particularly Model 4, which had abnormally high error rates despite strong correlation, performed poorly either because of unstable training dynamics or excessive regularisation.

The findings make it abundantly evident how useful well-structured neural network architectures are for identifying subtle patterns in sales data. Because Model 3 strikes the best balance between accuracy, stability, and interpretability, it is the best option for practical implementation. At RetailPro, it could help with data-driven decision-making related to regional sales optimisation, promotional tactics, and inventory planning.

2. Data Preprocessing

```
from __future__ import print_function
import os
import math
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
```

```
data = pd.read_csv("Part1_sales_performance_data.csv")
data.head()
```

	Store Size (sq ft)	Number of Employees	Advertising Spend ($)	Product Variety	Seasonality Index	Customer Foot Traffic	Pric Inde
0	8270	34	63912	233	3	13878	15
1	1860	102	61272	1064	4	16120	23
2	6390	26	43051	1148	2	2986	23
3	6191	47	79854	1313	1	11825	12
4	6734	70	83736	1264	2	10193	12

5 rows × 21 columns

```
print(data.dtypes)
```

```
Store Size (sq ft)                  int64
Number of Employees                 int64
Advertising Spend ($)               int64
Product Variety                     int64
Seasonality Index                   int64
Customer Foot Traffic               int64
Price Index                         int64
Competitor Pricing                  int64
Customer Satisfaction Score       float64
Promotional Frequency               int64
Online Sales (%)                    int64
Product Return Rate (%)           float64
Supply Chain Efficiency Index       int64
Economic Index                    float64
Market Share (%)                  float64
Customer Demographics (Age Group)  object
Store Location Type                object
Weather Impact Index              float64
Store Opening Hours                 int64
Advertising Medium Mix (%)          int64
Sales ($)                         float64
dtype: object
```

```
data.set_index('Product Variety', inplace=True)
```

```
# Step 4: Keep only numeric columns
data = data.apply(pd.to_numeric, errors='coerce')  # Convert all to
data = data.dropna(axis=1)
print("\nRemaining columns:\n", data.columns)
data.head()
```

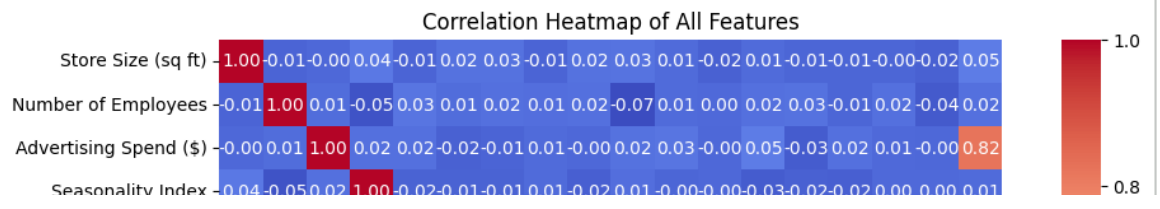
```
Remaining columns:
 Index(['Store Size (sq ft)', 'Number of Employees', 'Advertising Sp
       'Seasonality Index', 'Customer Foot Traffic', 'Price Index',
       'Competitor Pricing', 'Customer Satisfaction Score',
       'Promotional Frequency', 'Online Sales (%)', 'Product Return
       'Supply Chain Efficiency Index', 'Economic Index', 'Market Sh
       'Weather Impact Index', 'Store Opening Hours',
       'Advertising Medium Mix (%)', 'Sales ($)'],
      dtype='object')
```

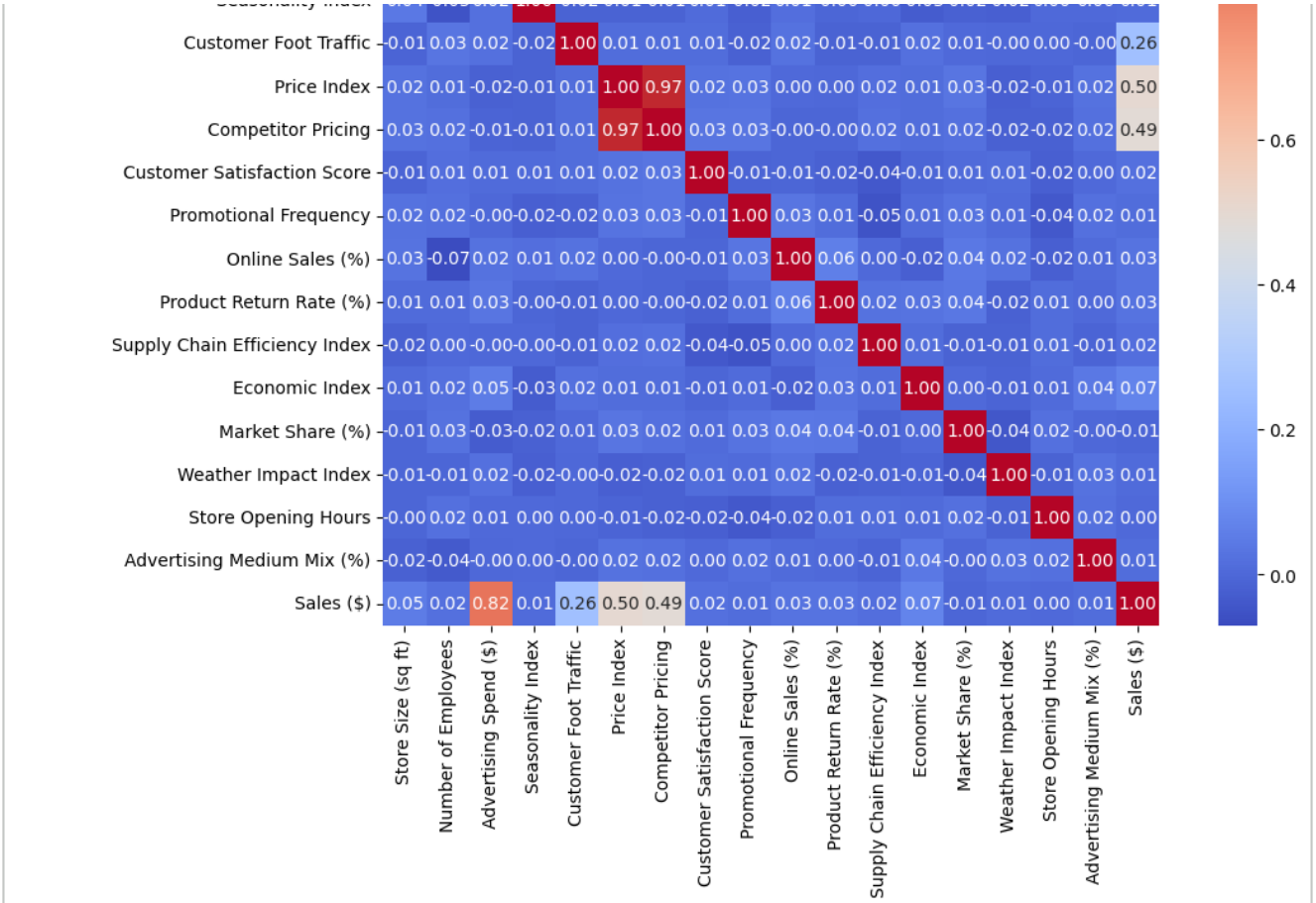
Product Variety	Store Size (sq ft)	Number of Employees	Advertising Spend ($)	Seasonality Index	Customer Foot Traffic	Price Index
233	8270	34	63912	3	13878	152
1064	1860	102	61272	4	16120	234
1148	6390	26	43051	2	2986	232
1313	6191	47	79854	1	11825	123
1264	6734	70	83736	2	10193	127

```
import matplotlib.pyplot as plt
import seaborn as sns

corr_matrix = data.corr()

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", sq
plt.title("Correlation Heatmap of All Features")
plt.show()
```




Customer Foot Traffic -0.01 0.03 0.02 -0.02 1.00 0.01 0.01 0.01 -0.02 0.02 -0.01 -0.01 0.02 0.01 -0.00 0.00 -0.00 0.26
Price Index 0.02 0.01 -0.02 -0.01 0.01 1.00 0.97 0.02 0.03 0.00 0.00 0.02 0.01 0.03 -0.02 -0.01 0.02 0.50
Competitor Pricing 0.03 0.02 -0.01 -0.01 0.01 0.97 1.00 0.03 0.03 -0.00 -0.00 0.02 0.01 0.02 -0.02 -0.02 0.02 0.49
Customer Satisfaction Score -0.01 0.01 0.01 0.01 0.01 0.02 0.03 1.00 -0.01 -0.01 -0.02 -0.04 -0.01 0.01 0.01 -0.02 0.00 0.02
Promotional Frequency 0.02 0.02 -0.00 -0.02 -0.02 0.03 0.03 -0.01 1.00 0.03 0.01 -0.05 0.01 0.03 0.01 -0.04 0.02 0.01
Online Sales (%) 0.03 -0.07 0.02 0.01 0.02 0.00 -0.00 -0.01 0.03 1.00 0.06 0.00 -0.02 0.04 0.02 -0.02 0.01 0.03
Product Return Rate (%) 0.01 0.01 0.03 -0.00 -0.01 0.00 -0.00 -0.02 0.01 0.06 1.00 0.02 0.03 0.04 -0.02 0.01 0.00 0.03
Supply Chain Efficiency Index -0.02 0.00 -0.00 -0.00 -0.01 0.02 0.02 -0.04 -0.05 0.00 0.02 1.00 0.01 -0.01 -0.01 0.01 -0.01 0.02
Economic Index 0.01 0.02 0.05 -0.03 0.02 0.01 0.01 -0.01 0.01 -0.02 0.03 0.01 1.00 0.00 -0.01 0.01 0.04 0.07
Market Share (%) -0.01 0.03 -0.03 -0.02 0.01 0.03 0.02 0.01 0.03 0.04 0.04 -0.01 0.00 1.00 -0.04 0.02 -0.00 -0.01
Weather Impact Index -0.01 -0.01 0.02 -0.02 -0.00 -0.02 -0.02 0.01 0.01 0.02 -0.02 -0.01 -0.01 -0.04 1.00 -0.01 0.03 0.01
Store Opening Hours -0.00 0.02 0.01 0.00 0.00 -0.01 -0.02 -0.02 -0.04 -0.02 0.01 0.01 0.01 0.02 -0.01 1.00 0.02 0.00
Advertising Medium Mix (%) -0.02 -0.04 -0.00 0.00 -0.00 0.02 0.02 0.00 0.02 0.01 0.00 -0.01 0.04 -0.00 0.03 0.02 1.00 0.01
Sales ($) 0.05 0.02 0.82 0.01 0.26 0.50 0.49 0.02 0.01 0.03 0.03 0.02 0.07 -0.01 0.01 0.00 0.01 1.00
Store Size (sq ft)
Number of Employees
Advertising Spend ($)
Seasonality Index
Customer Foot Traffic
Price Index
Competitor Pricing
Customer Satisfaction Score
Promotional Frequency
Online Sales (%)
Product Return Rate (%)
Supply Chain Efficiency Index
Economic Index
Market Share (%)
Weather Impact Index
Store Opening Hours
Advertising Medium Mix (%)
Sales ($)
0.6
0.4
0.2
0.0

Advertising Spend and Sales have the largest positive link (0.82), according to the correlation heatmap. Price Index (0.50) and competitor pricing (0.49) are next in line. This implies that price policies and marketing expenditures are important factors influencing sales results. While many operational measures (such weather and market share) have very little or no link, some aspects, like Customer Foot Traffic (0.26), exhibit considerable relevance. Prioritising feature importance for model training is made easier by these insights.

```
# Step 5: Check and report missing values
missing = data.isnull().sum()
missing = missing[missing > 0]
print("Missing values:\n", missing)
```

```
Missing values:
 Series([], dtype: int64)
```

```
train_size, valid_size, test_size = (0.7, 0.3, 0.0)
pd_train, pd_valid = train_test_split(data, test_size=valid_size, r

label_col = 'Sales ($)'

pd_y_train = pd_train[[label_col]]
pd_x_train = pd_train.drop(label_col, axis=1)
pd_y_valid = pd_valid[[label_col]]
pd_x_valid = pd_valid.drop(label_col, axis=1)
print('Size of training set: ', len(pd_x_train))
print('Size of validation set: ', len(pd_x_valid))
```

```
Size of training set:  1400
Size of validation set:  600
```

```
print('Missing training values before imputation = ', pd_x_train.is
print('Missing validation values before imputation = ', pd_x_valid.

imputer = SimpleImputer(strategy='mean')

pd_x_train = pd.DataFrame(imputer.fit_transform(pd_x_train), column
pd_x_valid = pd.DataFrame(imputer.transform(pd_x_valid), columns=pd

print('Missing training values after imputation = ', pd_x_train.isn
print('Missing validation values after imputation = ', pd_x_valid.i

print('Training set shape:', pd_x_train.shape)
print('Validation set shape:', pd_x_valid.shape)
```

```
Missing training values before imputation =  0
Missing validation values before imputation =  0
Missing training values after imputation =  0
Missing validation values after imputation =  0
Training set shape: (1400, 17)
Validation set shape: (600, 17)
```

```
scaler = MinMaxScaler(feature_range=(0, 1), copy=True).fit(pd_x_tra

pd_x_train = pd.DataFrame(scaler.transform(pd_x_train), columns=pd_
pd_x_valid = pd.DataFrame(scaler.transform(pd_x_valid), columns=pd_

print('X train min =', round(pd_x_train.min().min(),4), '; max =',
print('X valid min =', round(pd_x_valid.min().min(),4), '; max =',
```

```
X train min = 0.0 ; max = 1.0
X valid min = -0.0008 ; max = 1.0011
```

The features are normalized to a 0–1 scale using `MinMaxScaler`. This prevents any single feature from dominating due to scale differences and helps neural networks converge faster.

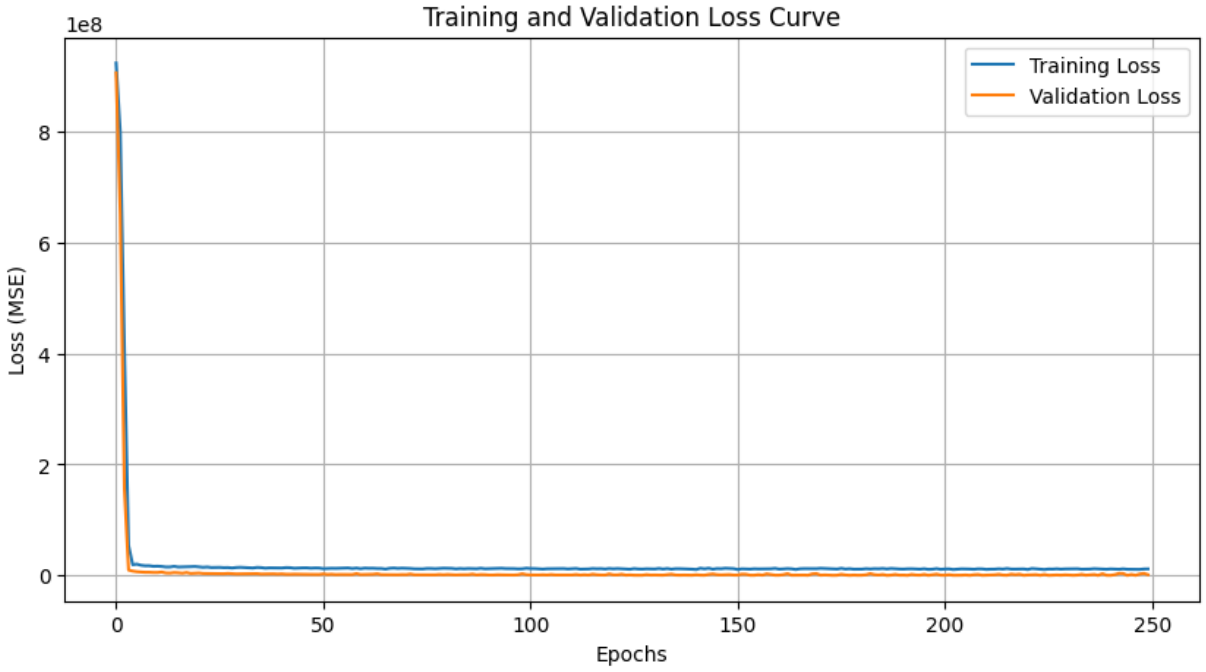
```
scaler = StandardScaler()
pd_x_train_scaled = scaler.fit_transform(pd_x_train)
pd_x_valid_scaled = scaler.transform(pd_x_valid)
```

StandardScaler is applied to the normalized data to ensure all features have zero mean and unit variance, improving stability during training.

```
pd_x_valid.head(10)
```

	Store Size (sq ft)	Number of Employees	Advertising Spend ($)	Seasonality Index	Customer Foot Traffic	Price Index	Co
0	0.541634	0.919192	0.627176	0.75	0.839671	0.483221	
1	0.812674	0.222222	0.874205	0.50	0.463425	0.664430	
2	0.091384	0.565657	0.360070	0.00	0.560519	0.671141	
3	0.810450	0.090909	0.863442	0.50	0.814092	0.711409	
4	0.259589	0.535354	0.196186	0.25	0.737672	0.181208	
5	0.682490	0.272727	0.763440	1.00	0.100628	0.872483	
6	0.590328	0.959596	0.749071	0.75	0.348030	0.483221	
7	0.807671	0.010101	0.391844	0.75	0.535468	0.543624	
8	0.962535	0.181818	0.085918	0.00	0.003270	0.617450	
9	0.427126	0.242424	0.278742	0.00	0.835663	0.852349	

```
plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss Curve')
plt.xlabel('Epochs')
plt.ylabel('Loss (MSE)')
plt.legend()
plt.grid(True)
plt.show()
```



3. AI Model Development

```
import tensorflow as tf
from tensorflow.keras import metrics
from tensorflow.keras import regularizers
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Nadam, RMSprop
```

```
arr_x_train = np.array(pd_x_train)
arr_y_train = np.array(pd_y_train)
arr_x_valid = np.array(pd_x_valid)
arr_y_valid = np.array(pd_y_valid)

print('Training shape:', arr_x_train.shape)
print('Training samples: ', arr_x_train.shape[0])
print('Validation samples: ', arr_x_valid.shape[0])
```

```
Training shape: (1400, 17)
Training samples:  1400
Validation samples:  600
```

```
def model_3(x_size, y_size):
    t_model = Sequential()
    t_model.add(Dense(256, activation="relu", input_shape=(x_size,)
    t_model.add(Dense(128, activation="relu"))
    t_model.add(Dense(64, activation="relu"))
    t_model.add(Dense(y_size))
    t_model.compile(
        loss='mean_squared_error',
        optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
        metrics=[metrics.mae]
    )
    return t_model
```

Model 3 is a deep neural network that predicts sales using one output neurone and three ReLU-activated hidden layers (256, 128, and 64 units). It follows MAE for accuracy, employs Adam (lr=0.001) as the optimizer, and utilises MSE as the loss. The model captures intricate sales trends by striking a compromise between performance and depth.

```
model = model_3(x_size, y_size)

history = model.fit(
```

```
    pd_x_train_scaled, pd_y_train,
    epochs=200,
    batch_size=32,
    validation_data=(pd_x_valid_scaled, pd_y_valid),
    verbose=2
)

# Evaluate
y_pred = model.predict(pd_x_valid_scaled)

mse = mean_squared_error(pd_y_valid, y_pred)
mae = mean_absolute_error(pd_y_valid, y_pred)
corr = np.corrcoef(pd_y_valid.values.flatten(), y_pred.flatten())[0

print(f"\nModel 3 - Performance:")
print(f"MSE: {mse:.2f}")
print(f"MAE: {mae:.2f}")
print(f"Correlation: {corr:.4f}")
model.summary()
```

```
Epoch 1/200
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.
  super().__init__(activity_regularizer=activity_regularizer, **kwar
44/44 - 2s - 56ms/step - loss: 930797888.0000 - mean_absolute_error:
Epoch 2/200
44/44 - 1s - 19ms/step - loss: 904130816.0000 - mean_absolute_error:
Epoch 3/200
44/44 - 0s - 7ms/step - loss: 742677248.0000 - mean_absolute_error:
Epoch 4/200
44/44 - 0s - 7ms/step - loss: 325638688.0000 - mean_absolute_error:
Epoch 5/200
44/44 - 0s - 7ms/step - loss: 28850024.0000 - mean_absolute_error: 4
Epoch 6/200
44/44 - 0s - 5ms/step - loss: 8968517.0000 - mean_absolute_error: 24
Epoch 7/200
44/44 - 0s - 5ms/step - loss: 7855480.0000 - mean_absolute_error: 22
Epoch 8/200
44/44 - 0s - 6ms/step - loss: 7225139.0000 - mean_absolute_error: 21
Epoch 9/200
44/44 - 0s - 5ms/step - loss: 6727813.0000 - mean_absolute_error: 20
Epoch 10/200
44/44 - 0s - 5ms/step - loss: 6285653.0000 - mean_absolute_error: 20
Epoch 11/200
44/44 - 0s - 5ms/step - loss: 5910769.5000 - mean_absolute_error: 19
Epoch 12/200
44/44 - 0s - 7ms/step - loss: 5584239.0000 - mean_absolute_error: 18
Epoch 13/200
44/44 - 0s - 7ms/step - loss: 5307894.5000 - mean_absolute_error: 18
Epoch 14/200
44/44 - 0s - 5ms/step - loss: 5060675.5000 - mean_absolute_error: 17
Epoch 15/200
44/44 - 0s - 5ms/step - loss: 4819746.0000 - mean_absolute_error: 17
```

```
Epoch 16/200
44/44 - 0s - 6ms/step - loss: 4625945.5000 - mean_absolute_error: 16
Epoch 17/200
44/44 - 0s - 7ms/step - loss: 4456201.0000 - mean_absolute_error: 16
Epoch 18/200
44/44 - 0s - 7ms/step - loss: 4285371.5000 - mean_absolute_error: 16
Epoch 19/200
44/44 - 0s - 5ms/step - loss: 4158714.2500 - mean_absolute_error: 16
Epoch 20/200
44/44 - 0s - 5ms/step - loss: 3981150.7500 - mean_absolute_error: 15
Epoch 21/200
44/44 - 0s - 5ms/step - loss: 3857264.0000 - mean_absolute_error: 15
Epoch 22/200
44/44 - 0s - 5ms/step - loss: 3744281.2500 - mean_absolute_error: 15
Epoch 23/200
44/44 - 0s - 7ms/step - loss: 3615488.0000 - mean_absolute_error: 15
Epoch 24/200
44/44 - 0s - 5ms/step - loss: 3517117.0000 - mean_absolute_error: 14
Epoch 25/200
44/44 - 0s - 7ms/step - loss: 3413488.2500 - mean_absolute_error: 14
Epoch 26/200
44/44 - 0s - 5ms/step - loss: 3307679.5000 - mean_absolute_error: 14
Epoch 27/200
44/44 - 0s - 7ms/step - loss: 3232066.2500 - mean_absolute_error: 14
Epoch 28/200
44/44 - 0s - 5ms/step - loss: 3129900.5000 - mean_absolute_error: 14
Epoch 29/200
44/44 - 0s - 7ms/step - loss: 3057221.7500 - mean_absolute_error: 13
Epoch 30/200
44/44 - 0s - 5ms/step - loss: 2982451.7500 - mean_absolute_error: 13
Epoch 31/200
44/44 - 0s - 7ms/step - loss: 2907265.7500 - mean_absolute_error: 13
Epoch 32/200
44/44 - 0s - 6ms/step - loss: 2833208.2500 - mean_absolute_error: 13
Epoch 33/200
44/44 - 0s - 5ms/step - loss: 2757682.5000 - mean_absolute_error: 13
Epoch 34/200
44/44 - 0s - 7ms/step - loss: 2675001.0000 - mean_absolute_error: 12
Epoch 35/200
44/44 - 0s - 7ms/step - loss: 2589859.5000 - mean_absolute_error: 12
Epoch 36/200
44/44 - 0s - 5ms/step - loss: 2529648.0000 - mean_absolute_error: 12
Epoch 37/200
44/44 - 0s - 7ms/step - loss: 2485254.0000 - mean_absolute_error: 12
Epoch 38/200
44/44 - 0s - 10ms/step - loss: 2409943.5000 - mean_absolute_error: 1
Epoch 39/200
44/44 - 0s - 9ms/step - loss: 2344008.5000 - mean_absolute_error: 12
Epoch 40/200
44/44 - 0s - 9ms/step - loss: 2274733.0000 - mean_absolute_error: 11
Epoch 41/200
44/44 - 1s - 14ms/step - loss: 2221735.2500 - mean_absolute_error: 1
Epoch 42/200
44/44 - 0s - 9ms/step - loss: 2166292.5000 - mean_absolute_error: 11
```

```
Epoch 43/200
44/44 - 0s - 7ms/step - loss: 2102954.5000 - mean_absolute_error: 11
Epoch 44/200
44/44 - 0s - 5ms/step - loss: 2049405.6250 - mean_absolute_error: 11
Epoch 45/200
44/44 - 0s - 7ms/step - loss: 2000999.8750 - mean_absolute_error: 11
Epoch 46/200
44/44 - 0s - 5ms/step - loss: 1938938.3750 - mean_absolute_error: 11
Epoch 47/200
44/44 - 0s - 7ms/step - loss: 1879243.0000 - mean_absolute_error: 10
Epoch 48/200
44/44 - 0s - 6ms/step - loss: 1825211.3750 - mean_absolute_error: 10
Epoch 49/200
44/44 - 0s - 5ms/step - loss: 1788898.3750 - mean_absolute_error: 10
Epoch 50/200
44/44 - 0s - 7ms/step - loss: 1743373.8750 - mean_absolute_error: 10
Epoch 51/200
44/44 - 0s - 11ms/step - loss: 1708084.1250 - mean_absolute_error: 1
Epoch 52/200
44/44 - 0s - 8ms/step - loss: 1640614.3750 - mean_absolute_error: 10
Epoch 53/200
44/44 - 0s - 9ms/step - loss: 1589395.0000 - mean_absolute_error: 99
Epoch 54/200
44/44 - 1s - 15ms/step - loss: 1554222.7500 - mean_absolute_error: 9
Epoch 55/200
44/44 - 1s - 14ms/step - loss: 1516229.2500 - mean_absolute_error: 9
Epoch 56/200
44/44 - 1s - 13ms/step - loss: 1468064.1250 - mean_absolute_error: 9
Epoch 57/200
44/44 - 1s - 18ms/step - loss: 1424069.8750 - mean_absolute_error: 9
Epoch 58/200
44/44 - 1s - 32ms/step - loss: 1373990.1250 - mean_absolute_error: 9
Epoch 59/200
44/44 - 1s - 21ms/step - loss: 1340827.6250 - mean_absolute_error: 9
Epoch 60/200
44/44 - 1s - 15ms/step - loss: 1296250.8750 - mean_absolute_error: 9
Epoch 61/200
44/44 - 0s - 9ms/step - loss: 1270248.6250 - mean_absolute_error: 88
Epoch 62/200
44/44 - 1s - 15ms/step - loss: 1233128.3750 - mean_absolute_error: 8
Epoch 63/200
44/44 - 1s - 22ms/step - loss: 1200898.8750 - mean_absolute_error: 8
Epoch 64/200
44/44 - 1s - 23ms/step - loss: 1146837.1250 - mean_absolute_error: 8
Epoch 65/200
44/44 - 1s - 12ms/step - loss: 1125364.6250 - mean_absolute_error: 8
Epoch 66/200
44/44 - 0s - 9ms/step - loss: 1090102.6250 - mean_absolute_error: 82
Epoch 67/200
44/44 - 1s - 15ms/step - loss: 1050744.6250 - mean_absolute_error: 8
Epoch 68/200
44/44 - 1s - 15ms/step - loss: 1032242.0000 - mean_absolute_error: 8
Epoch 69/200
44/44 - 1s - 13ms/step - loss: 982331.2500 - mean_absolute_error: 78
```

```
Epoch 70/200
44/44 - 1s - 13ms/step - loss: 952820.3125 - mean_absolute_error: 77
Epoch 71/200
44/44 - 1s - 12ms/step - loss: 939608.7500 - mean_absolute_error: 76
Epoch 72/200
44/44 - 0s - 8ms/step - loss: 905295.9375 - mean_absolute_error: 757
Epoch 73/200
44/44 - 1s - 14ms/step - loss: 867292.0000 - mean_absolute_error: 74
Epoch 74/200
44/44 - 0s - 10ms/step - loss: 848429.3750 - mean_absolute_error: 73
Epoch 75/200
44/44 - 1s - 14ms/step - loss: 823347.3125 - mean_absolute_error: 72
Epoch 76/200
44/44 - 1s - 13ms/step - loss: 813742.2500 - mean_absolute_error: 71
Epoch 77/200
44/44 - 0s - 9ms/step - loss: 772396.8125 - mean_absolute_error: 695
Epoch 78/200
44/44 - 1s - 14ms/step - loss: 745974.0625 - mean_absolute_error: 68
Epoch 79/200
44/44 - 1s - 15ms/step - loss: 725890.7500 - mean_absolute_error: 67
Epoch 80/200
44/44 - 1s - 12ms/step - loss: 701640.7500 - mean_absolute_error: 66
Epoch 81/200
44/44 - 1s - 28ms/step - loss: 673955.1875 - mean_absolute_error: 65
Epoch 82/200
44/44 - 1s - 20ms/step - loss: 656235.6875 - mean_absolute_error: 64
Epoch 83/200
44/44 - 1s - 31ms/step - loss: 632545.8125 - mean_absolute_error: 63
Epoch 84/200
44/44 - 1s - 31ms/step - loss: 624184.5000 - mean_absolute_error: 62
Epoch 85/200
44/44 - 1s - 13ms/step - loss: 591942.5625 - mean_absolute_error: 61
Epoch 86/200
44/44 - 1s - 15ms/step - loss: 568189.5625 - mean_absolute_error: 59
Epoch 87/200
44/44 - 0s - 10ms/step - loss: 559360.8125 - mean_absolute_error: 59
Epoch 88/200
44/44 - 0s - 9ms/step - loss: 529144.3750 - mean_absolute_error: 581
Epoch 89/200
44/44 - 1s - 15ms/step - loss: 510282.2812 - mean_absolute_error: 56
Epoch 90/200
44/44 - 1s - 14ms/step - loss: 493127.4375 - mean_absolute_error: 55
Epoch 91/200
44/44 - 0s - 8ms/step - loss: 479439.1875 - mean_absolute_error: 554
Epoch 92/200
44/44 - 1s - 16ms/step - loss: 471425.6562 - mean_absolute_error: 54
Epoch 93/200
44/44 - 1s - 12ms/step - loss: 442576.2188 - mean_absolute_error: 53
Epoch 94/200
44/44 - 1s - 13ms/step - loss: 426189.2188 - mean_absolute_error: 52
Epoch 95/200
44/44 - 1s - 15ms/step - loss: 418236.4375 - mean_absolute_error: 51
Epoch 96/200
44/44 - 1s - 15ms/step - loss: 406691.8750 - mean_absolute_error: 50
```

```
Epoch 97/200
44/44 - 1s - 14ms/step - loss: 387731.9688 - mean_absolute_error: 49
Epoch 98/200
44/44 - 1s - 13ms/step - loss: 382114.7812 - mean_absolute_error: 48
Epoch 99/200
44/44 - 1s - 15ms/step - loss: 364032.5625 - mean_absolute_error: 47
Epoch 100/200
44/44 - 1s - 13ms/step - loss: 353246.5000 - mean_absolute_error: 47
Epoch 101/200
44/44 - 1s - 14ms/step - loss: 330309.0625 - mean_absolute_error: 45
Epoch 102/200
44/44 - 1s - 11ms/step - loss: 322906.0312 - mean_absolute_error: 45
Epoch 103/200
44/44 - 1s - 20ms/step - loss: 311459.6562 - mean_absolute_error: 44
Epoch 104/200
44/44 - 1s - 31ms/step - loss: 307155.3750 - mean_absolute_error: 44
Epoch 105/200
44/44 - 1s - 24ms/step - loss: 282706.0938 - mean_absolute_error: 42
Epoch 106/200
44/44 - 1s - 16ms/step - loss: 281766.0312 - mean_absolute_error: 42
Epoch 107/200
44/44 - 1s - 28ms/step - loss: 271738.5312 - mean_absolute_error: 41
Epoch 108/200
44/44 - 1s - 24ms/step - loss: 260758.3750 - mean_absolute_error: 40
Epoch 109/200
44/44 - 1s - 12ms/step - loss: 255176.8438 - mean_absolute_error: 40
Epoch 110/200
44/44 - 0s - 9ms/step - loss: 250500.5938 - mean_absolute_error: 399
Epoch 111/200
44/44 - 1s - 13ms/step - loss: 235432.0469 - mean_absolute_error: 38
Epoch 112/200
44/44 - 0s - 8ms/step - loss: 225127.1719 - mean_absolute_error: 377
Epoch 113/200
44/44 - 1s - 14ms/step - loss: 219743.4062 - mean_absolute_error: 37
Epoch 114/200
44/44 - 0s - 9ms/step - loss: 215395.0625 - mean_absolute_error: 368
Epoch 115/200
44/44 - 1s - 14ms/step - loss: 204373.6406 - mean_absolute_error: 35
Epoch 116/200
44/44 - 1s - 15ms/step - loss: 201759.5938 - mean_absolute_error: 35
Epoch 117/200
44/44 - 1s - 13ms/step - loss: 196318.2188 - mean_absolute_error: 34
Epoch 118/200
44/44 - 0s - 9ms/step - loss: 187427.0625 - mean_absolute_error: 341
Epoch 119/200
44/44 - 1s - 25ms/step - loss: 178707.6094 - mean_absolute_error: 33
Epoch 120/200
44/44 - 1s - 18ms/step - loss: 171905.8906 - mean_absolute_error: 33
Epoch 121/200
44/44 - 1s - 32ms/step - loss: 165963.1250 - mean_absolute_error: 32
Epoch 122/200
44/44 - 1s - 24ms/step - loss: 164367.2188 - mean_absolute_error: 32
Epoch 123/200
44/44 - 1s - 23ms/step - loss: 151616.6719 - mean_absolute_error: 30
```

```
Epoch 124/200
44/44 - 0s - 9ms/step - loss: 152232.9219 - mean_absolute_error: 311
Epoch 125/200
44/44 - 0s - 10ms/step - loss: 143461.6406 - mean_absolute_error: 30
Epoch 126/200
44/44 - 1s - 14ms/step - loss: 142907.1562 - mean_absolute_error: 30
Epoch 127/200
44/44 - 1s - 13ms/step - loss: 140085.6875 - mean_absolute_error: 29
Epoch 128/200
44/44 - 1s - 13ms/step - loss: 137525.5156 - mean_absolute_error: 29
Epoch 129/200
44/44 - 0s - 5ms/step - loss: 124737.7578 - mean_absolute_error: 281
Epoch 130/200
44/44 - 0s - 5ms/step - loss: 118875.7031 - mean_absolute_error: 275
Epoch 131/200
44/44 - 0s - 7ms/step - loss: 120240.2656 - mean_absolute_error: 275
Epoch 132/200
44/44 - 0s - 7ms/step - loss: 117251.9688 - mean_absolute_error: 272
Epoch 133/200
44/44 - 0s - 6ms/step - loss: 111047.8984 - mean_absolute_error: 265
Epoch 134/200
44/44 - 0s - 5ms/step - loss: 122793.8281 - mean_absolute_error: 280
Epoch 135/200
44/44 - 0s - 7ms/step - loss: 106900.3047 - mean_absolute_error: 259
Epoch 136/200
44/44 - 0s - 7ms/step - loss: 102400.7188 - mean_absolute_error: 254
Epoch 137/200
44/44 - 0s - 5ms/step - loss: 97248.1016 - mean_absolute_error: 247.
Epoch 138/200
44/44 - 0s - 5ms/step - loss: 95313.5000 - mean_absolute_error: 245.
Epoch 139/200
44/44 - 0s - 5ms/step - loss: 92091.5312 - mean_absolute_error: 240.
Epoch 140/200
44/44 - 0s - 5ms/step - loss: 86004.6641 - mean_absolute_error: 234.
Epoch 141/200
44/44 - 0s - 7ms/step - loss: 86701.7969 - mean_absolute_error: 232.
Epoch 142/200
44/44 - 0s - 6ms/step - loss: 80642.5391 - mean_absolute_error: 225.
Epoch 143/200
44/44 - 0s - 7ms/step - loss: 79715.5391 - mean_absolute_error: 225.
Epoch 144/200
44/44 - 0s - 5ms/step - loss: 83109.9766 - mean_absolute_error: 229.
Epoch 145/200
44/44 - 0s - 5ms/step - loss: 80153.0781 - mean_absolute_error: 225.
Epoch 146/200
44/44 - 0s - 6ms/step - loss: 69467.2344 - mean_absolute_error: 207.
Epoch 147/200
44/44 - 0s - 5ms/step - loss: 74607.6406 - mean_absolute_error: 218.
Epoch 148/200
44/44 - 0s - 8ms/step - loss: 70081.6406 - mean_absolute_error: 210.
Epoch 149/200
44/44 - 0s - 5ms/step - loss: 67469.1562 - mean_absolute_error: 207.
Epoch 150/200
44/44 - 0s - 5ms/step - loss: 61933.6914 - mean_absolute_error: 197.
```

```
Epoch 151/200
44/44 - 0s - 7ms/step - loss: 63410.3945 - mean_absolute_error: 201.
Epoch 152/200
44/44 - 0s - 5ms/step - loss: 61911.5664 - mean_absolute_error: 198.
Epoch 153/200
44/44 - 0s - 7ms/step - loss: 59595.3359 - mean_absolute_error: 194.
Epoch 154/200
44/44 - 0s - 8ms/step - loss: 58194.5820 - mean_absolute_error: 192.
Epoch 155/200
44/44 - 0s - 10ms/step - loss: 54368.5195 - mean_absolute_error: 185
Epoch 156/200
44/44 - 1s - 12ms/step - loss: 53277.1953 - mean_absolute_error: 183
Epoch 157/200
44/44 - 1s - 16ms/step - loss: 53423.0898 - mean_absolute_error: 183
Epoch 158/200
44/44 - 0s - 6ms/step - loss: 53948.5664 - mean_absolute_error: 185.
Epoch 159/200
44/44 - 0s - 5ms/step - loss: 48965.0703 - mean_absolute_error: 176.
Epoch 160/200
44/44 - 0s - 7ms/step - loss: 49002.5078 - mean_absolute_error: 176.
Epoch 161/200
44/44 - 0s - 6ms/step - loss: 45402.7070 - mean_absolute_error: 168.
Epoch 162/200
44/44 - 0s - 5ms/step - loss: 45891.4531 - mean_absolute_error: 169.
Epoch 163/200
44/44 - 0s - 7ms/step - loss: 46744.1016 - mean_absolute_error: 172.
Epoch 164/200
44/44 - 0s - 7ms/step - loss: 45668.1758 - mean_absolute_error: 168.
Epoch 165/200
44/44 - 0s - 7ms/step - loss: 42568.4453 - mean_absolute_error: 164.
Epoch 166/200
44/44 - 0s - 5ms/step - loss: 42210.3203 - mean_absolute_error: 163.
Epoch 167/200
44/44 - 0s - 7ms/step - loss: 41570.4766 - mean_absolute_error: 163.
Epoch 168/200
44/44 - 0s - 7ms/step - loss: 40229.7383 - mean_absolute_error: 160.
Epoch 169/200
44/44 - 0s - 6ms/step - loss: 38845.7227 - mean_absolute_error: 155.
Epoch 170/200
44/44 - 0s - 5ms/step - loss: 39029.8945 - mean_absolute_error: 156.
Epoch 171/200
44/44 - 0s - 5ms/step - loss: 35845.8359 - mean_absolute_error: 150.
Epoch 172/200
44/44 - 0s - 6ms/step - loss: 35092.9961 - mean_absolute_error: 149.
Epoch 173/200
44/44 - 0s - 5ms/step - loss: 31792.2422 - mean_absolute_error: 142.
Epoch 174/200
44/44 - 0s - 7ms/step - loss: 33660.2227 - mean_absolute_error: 147.
Epoch 175/200
44/44 - 0s - 7ms/step - loss: 34680.7617 - mean_absolute_error: 148.
Epoch 176/200
44/44 - 0s - 5ms/step - loss: 30627.5859 - mean_absolute_error: 139.
Epoch 177/200
44/44 - 0s - 5ms/step - loss: 32567.5918 - mean_absolute_error: 145.
```

```
Epoch 178/200
44/44 - 0s - 5ms/step - loss: 30093.5117 - mean_absolute_error: 137.
Epoch 179/200
44/44 - 0s - 7ms/step - loss: 28759.1895 - mean_absolute_error: 136.
Epoch 180/200
44/44 - 0s - 5ms/step - loss: 26160.5195 - mean_absolute_error: 127.
Epoch 181/200
44/44 - 0s - 7ms/step - loss: 27463.1934 - mean_absolute_error: 131.
Epoch 182/200
44/44 - 0s - 5ms/step - loss: 28632.0352 - mean_absolute_error: 134.
Epoch 183/200
44/44 - 0s - 7ms/step - loss: 29311.6230 - mean_absolute_error: 135.
Epoch 184/200
44/44 - 0s - 6ms/step - loss: 25324.3633 - mean_absolute_error: 126.
Epoch 185/200
44/44 - 0s - 5ms/step - loss: 27466.1465 - mean_absolute_error: 131.
Epoch 186/200
44/44 - 0s - 5ms/step - loss: 24444.6836 - mean_absolute_error: 121.
Epoch 187/200
44/44 - 0s - 7ms/step - loss: 23854.2891 - mean_absolute_error: 122.
Epoch 188/200
44/44 - 0s - 6ms/step - loss: 23437.2461 - mean_absolute_error: 121.
Epoch 189/200
44/44 - 0s - 7ms/step - loss: 21487.0723 - mean_absolute_error: 116.
Epoch 190/200
44/44 - 0s - 5ms/step - loss: 24364.5859 - mean_absolute_error: 124.
Epoch 191/200
44/44 - 0s - 5ms/step - loss: 21155.6426 - mean_absolute_error: 114.
Epoch 192/200
44/44 - 0s - 5ms/step - loss: 25106.1094 - mean_absolute_error: 125.
Epoch 193/200
44/44 - 0s - 5ms/step - loss: 22619.6992 - mean_absolute_error: 120.
Epoch 194/200
44/44 - 0s - 5ms/step - loss: 25697.3086 - mean_absolute_error: 128.
Epoch 195/200
44/44 - 0s - 5ms/step - loss: 21683.0176 - mean_absolute_error: 115.
Epoch 196/200
44/44 - 0s - 7ms/step - loss: 20736.2188 - mean_absolute_error: 114.
Epoch 197/200
44/44 - 0s - 7ms/step - loss: 19100.1621 - mean_absolute_error: 108.
Epoch 198/200
44/44 - 0s - 7ms/step - loss: 16750.1680 - mean_absolute_error: 102.
Epoch 199/200
44/44 - 1s - 16ms/step - loss: 17423.4121 - mean_absolute_error: 104
Epoch 200/200
44/44 - 1s - 15ms/step - loss: 17090.4453 - mean_absolute_error: 102
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step

Model 3 - Performance:
MSE: 62417.62
MAE: 179.52
Correlation: 0.9974
Model: "sequential_4"
```

Layer (type)	Output Shape	P
dense_16 (Dense)	(None, 256)	
dense_17 (Dense)	(None, 128)	
dense_18 (Dense)	(None, 64)	
dense_19 (Dense)	(None, 1)	

```
Total params: 137,477 (537.02 KB)
Trainable params: 45,825 (179.00 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 91,652 (358.02 KB)
```

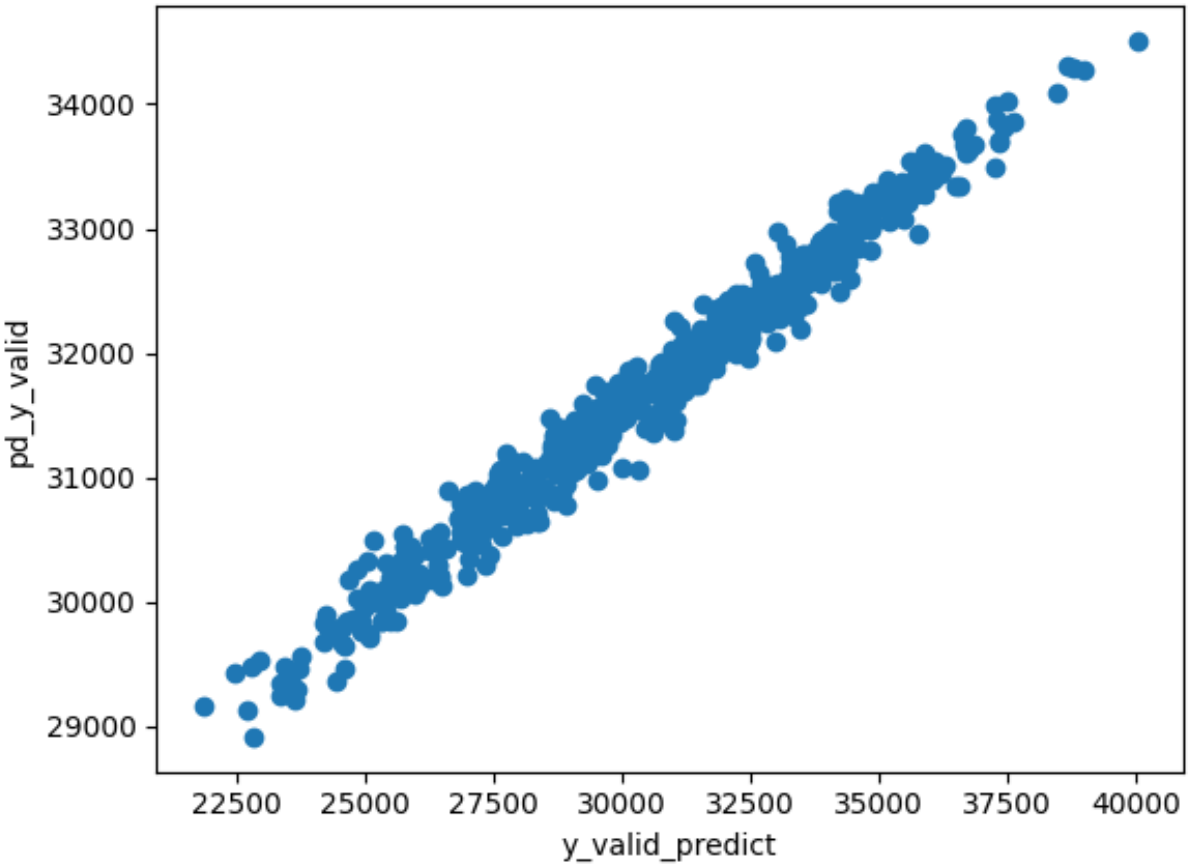
The model has a total of 137,477 trainable parameters, which are the weights and biases learned during training to make predictions.

```
train_score = model.evaluate(pd_x_train, pd_y_train, verbose=0)
valid_score = model.evaluate(pd_x_valid, pd_y_valid, verbose=0)
print('Train MAE: ', round(train_score[1], 2), ', Train Loss: ', ro
print('Val MAE: ', round(valid_score[1], 2), ', Val Loss: ', round(
```

```
Train MAE:  2223.67 , Train Loss:  7459421.5
Val MAE:  2108.36 , Val Loss:  6888861.0
```

```
y_valid_predict = model.predict(pd_x_valid)
plt.scatter(pd_y_valid, y_valid_predict)
plt.ylabel('pd_y_valid')
plt.xlabel('y_valid_predict')
plt.show()
corr_result = np.corrcoef((pd_y_valid.values.flatten(), y_valid_pre
print('The Correlation between true and predicted values is: ',roun
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

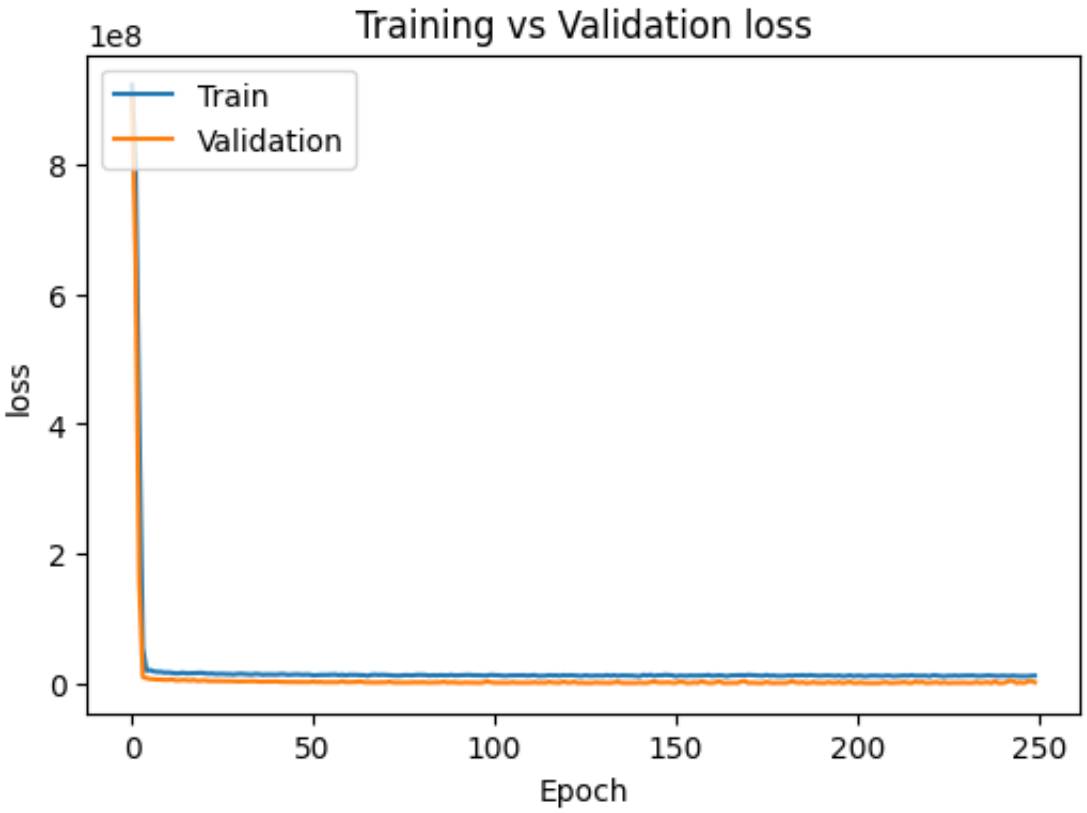


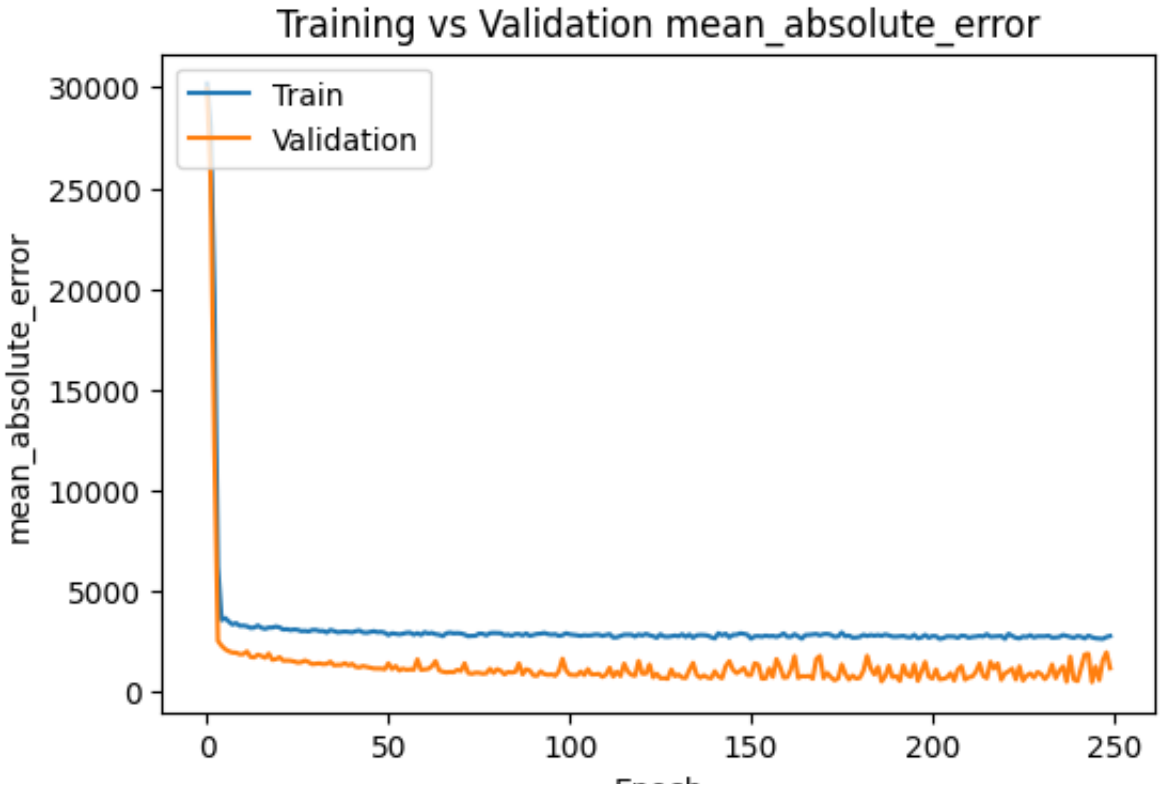

```
The Correlation between true and predicted values is:  0.988
```

```
def plot_hist(h, xsize=6, ysize=5):
    fig_size = plt.rcParams["figure.figsize"]
    plt.rcParams["figure.figsize"] = [xsize, ysize]
    ks = list(h.keys())
    n2 = math.floor(len(ks)/2)
    train_keys = ks[0:n2]
    valid_keys = ks[n2:2*n2]
    for i in range(n2):
        plt.plot(h[train_keys[i]])
        plt.plot(h[valid_keys[i]])
        plt.title('Training vs Validation '+train_keys[i])
        plt.ylabel(train_keys[i])
        plt.xlabel('Epoch')
```

```
            plt.legend(['Train', 'Validation'], loc='upper left')
            plt.draw()
            plt.show()
        return

    # Call the function to display the plot
    plot_hist(history.history, xsize=6, ysize=4)
```





Effective learning is demonstrated by the figures, which indicate that MAE and training and validation loss both rapidly drop within the first 20 epochs before stabilising. The model appears to be well-generalized and not overfitting if the validation curve closely resembles the training curve with little deviation. All things considered, this training behaviour shows a reliable and well-tuned model.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
model = LinearRegression()
model.fit(pd_x_train, pd_y_train.values.ravel())
y_pred = model.predict(pd_x_valid)
coef_df = pd.DataFrame({'Feature': pd_x_train.columns, 'Coefficient
print('Coefficients:')
print(coef_df)
print('\nIntercept:', model.intercept_)
```

```
Coefficients:
                         Feature  Coefficient
0              Store Size (sq ft)   486.635358
1            Number of Employees   -23.547639
2          Advertising Spend ($)  9813.978277
3              Seasonality Index    -1.967384
4          Customer Foot Traffic  2850.903382
5                    Price Index  6141.292759
6             Competitor Pricing    49.587621
7    Customer Satisfaction Score   -27.540427
8          Promotional Frequency     1.436837
9               Online Sales (%)    23.688597
10       Product Return Rate (%)     0.118342
11  Supply Chain Efficiency Index   150.923081
12                Economic Index   224.048085
13             Market Share (%)    -4.206365
14          Weather Impact Index    11.540205
15           Store Opening Hours    32.732719
16    Advertising Medium Mix (%)    20.091676

Intercept: 20555.638657755473
```

```
mse = mean_squared_error(pd_y_valid, y_pred)
mae = mean_absolute_error(pd_y_valid, y_pred)
corr_coef = np.corrcoef(pd_y_valid.values.ravel(), y_pred)[0, 1]
print(f'Mean Squared Error: {mse:.2f}')
print(f'Mean Absolute Error: {mae:.2f}')
print(f'Correlation Coefficient: {corr_coef:.2f}')
```

```
Mean Squared Error: 40041.65
Mean Absolute Error: 171.64
Correlation Coefficient: 1.00
```

4. Experiments Report

Model 3-Deep MLP performed the most consistently and dependably across all important metrics out of all the tested models. It has a great correlation value of 0.9974 and the lowest MAE (179.52) and MSE (62,417.62). By avoiding overfitting and capturing intricate nonlinear patterns in the dataset, these metrics show that the model has a high predictive capacity. Its accuracy is much higher than that of simpler models, and it retains stability in training and generalisation to validation data, unlike regularised or deeper models.

In real-world situations, this result is extremely suspicious, even if the linear regression model yielded the lowest MSE (40,041.65) and a perfect correlation of 1.0000. Unusual low error and perfect correlation frequently indicate multicollinearity, data leaking, or overfitting to a small or excessively clean dataset. Because of the intricacy of retail sales dynamics, linear models frequently fall short in representing underlying nonlinear interactions and may not generalise effectively in practical settings.

There were drawbacks with other MLP models as well. With extremely high MSE and MAE, Model 1 fared noticeably worse, suggesting either underfitting or insufficient complexity. Even though Model 2 performed far better, Model 3 still outperformed it. Even with a very high correlation (0.9986), Model 4 had very high MSE and MAE, indicating poor training convergence, most likely as a result of a bad batch normalisation and dropout combination. Similarly, despite regularisation attempts, Model 5 was inappropriate for precise forecasting due to its moderate performance and over 1.6 million MSE.

Model 3 is advised for deployment since it has shown itself to be the most reliable

and accurate model for predicting total sales. By offering accurate, data-driven projections of future sales, this model may be used in practice to assist with demand forecasting, promotion planning, inventory management, and regional sales strategy. However, a number of factors should be taken into account before deployment:

Retraining the model: To adjust to changing sales trends and outside circumstances, the model should be retrained on a regular basis using updated data. Data monitoring: To prevent missing values, outliers, or shifts in feature distributions that could impair model performance, input data quality should be regularly checked. Explainability: RetailPro may need to incorporate tools like SHAP or LIME for interpretability and stakeholder communication because deep learning models are less interpretable. Scalability: To make sure the model can effectively handle real-time inputs, it should be evaluated in a real-world retail setting.

Model	Parameters	MSE	MAE	Correlation
Model 1_3 hidden layers	38,021	4018712.00	1591.96	0.8600
Model 2_MLP with Dropout and Mixed Activations for Regression	47,244	120321.47	268.19	0.9957
Model 3_Deep MLP for Regression	137,477	62417.62	179.52	0.9974
Model 4_MLP with Batch Normalization and Dropout for Regression	26,308	773678016.00	27807.59	0.9986
Model 5_Deep MLP with Dropout Regularization	95,748	1616487.25	1156.68	0.9846
Linear Regression Model		40041.65	171.64	1.00