Introduction for the California Housing Price Prediction Project

In this project, we build and compare multiple machine learning models to predict housing prices in California. Using Python and libraries like Pandas, NumPy, Scikit-learn, and TensorFlow/Keras, to analyze California housing dataset.This dataset includes features such as geographical coordinates, housing attributes, and proximity to the ocean.The target variable is the median housing price in various districts.

The primary objective is to explore regression techniques to predict continuous values effectively. The project demonstrates the end-to-end machine learning workflow, including data preprocessing, feature scaling, model selection, and evaluation.

The goal of this project is to predict median house prices in California using various machine learning models. We experiment with traditional regression models and neural networks, comparing their performance to determine the best approach

IMPORTING NECESSARY LIBRARIES AND DEPENDENCIES

These libraries are essential for data handling (Pandas), numerical computations (NumPy), and data visualization (Matplotlib, Seaborn).Also We import machine learning models (LinearRegression, RandomForest, etc.), preprocessing tools (StandardScaler), and evaluation metrics (mean_squared_error) from Scikit-learn.While TensorFlow/Keras is used to design, train, and save neural networks.

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.metrics import mean_squared_error as mse
from sklearn.neighbors import KNeighborsRegressor
```

Loading the Dataset

In [ ]:

```
housing_pd = pd.read_csv('/content/housing.csv')
```

In [ ]:

```
housing_pd
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	media
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	
...	...	...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median
20637	-121.89	39.48	37.0	2254.0	485.0	1007.0	493.0	1.7000	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	

20640 rows × 10 columns

In [ ]:

```
#To understand the distribution of the categorical variable ocean_proximity.
housing_pd['ocean_proximity'].value_counts()
```

Out[ ]:

ocean_proximity	count
<1H OCEAN	9136
INLAND	6551
NEAR OCEAN	2658
NEAR BAY	2290
ISLAND	5

dtype: int64

Shuffling and Encoding categorical variables

Shuffling ensures randomness in the dataset. Encoding converts ocean_proximity into numerical "dummy variables," which machine learning models can process.

In [ ]:

```
housing_pd_shuffled = housing_pd.sample(n=len(housing_pd), random_state=1)
housing_pd_shuffled
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median
4712	-118.36	34.06	39.0	2810.0	670.0	1109.0	624.0	3.2500	
2151	-119.78	36.78	37.0	2185.0	455.0	1143.0	438.0	1.9784	
15927	-122.42	37.73	46.0	1819.0	411.0	1534.0	406.0	4.0132	
82	-122.28	37.81	52.0	340.0	97.0	200.0	87.0	1.5208	
8161	-118.13	33.82	37.0	1530.0	290.0	711.0	283.0	5.1795	
...	...	...	...	...	...	...	...	...	
10955	-117.88	33.76	17.0	1768.0	474.0	1079.0	436.0	1.7823	
17289	-119.63	34.42	42.0	1765.0	263.0	753.0	260.0	8.5608	
5192	-118.26	33.93	42.0	1433.0	295.0	775.0	293.0	1.1326	
12172	-117.16	33.73	10.0	2381.0	454.0	1323.0	477.0	2.6322	
235	-122.20	37.79	35.0	1802.0	459.0	1009.0	390.0	2.3036	

20640 rows × 10 columns

In [ ]:

```
# Preprocess ocean_proximity
housing_pd_shuffled = housing_pd.sample(n=len(housing_pd), random_state=1)
```

```
housing_pd_final = pd.concat(
    [housing_pd_shuffled.drop('ocean_proximity', axis=1),
     pd.get_dummies(housing_pd_shuffled['ocean_proximity'])],
    axis=1
)
```

In [ ]:

```
# Rearrange columns
housing_pd_final = housing_pd_final[[
    'longitude', 'latitude', 'housing_median_age', 'total_rooms',
    'total_bedrooms', 'population', 'households', 'median_income',
    '<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN', 'median_house_value']]
```

In [ ]:

```
housing_pd_final
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	<1H OCEAN
4712	-118.36	34.06	39.0	2810.0	670.0	1109.0	624.0	3.2500	True
2151	-119.78	36.78	37.0	2185.0	455.0	1143.0	438.0	1.9784	False
15927	-122.42	37.73	46.0	1819.0	411.0	1534.0	406.0	4.0132	False
82	-122.28	37.81	52.0	340.0	97.0	200.0	87.0	1.5208	False
8161	-118.13	33.82	37.0	1530.0	290.0	711.0	283.0	5.1795	True
...	...	...	...	...	...	...	...	...	...
10955	-117.88	33.76	17.0	1768.0	474.0	1079.0	436.0	1.7823	True
17289	-119.63	34.42	42.0	1765.0	263.0	753.0	260.0	8.5608	True
5192	-118.26	33.93	42.0	1433.0	295.0	775.0	293.0	1.1326	True
12172	-117.16	33.73	10.0	2381.0	454.0	1323.0	477.0	2.6322	False
235	-122.20	37.79	35.0	1802.0	459.0	1009.0	390.0	2.3036	False

20640 rows × 14 columns

Handeling Missing Values

In [ ]:

```
housing_pd_final = housing_pd_final.dropna()
```

In [ ]:

```
len(housing_pd_final)
```

Out[ ]:

```
20433
```

Splitting the data

Here Data is divided into:

1)Training set: Used to train the model.

2)Validation set: Used to fine-tune hyperparameters.

3)Test set: Used to evaluate the final model's performance.

In [ ]:

```
train_pd, test_pd, val_pd = housing_pd_final[:18000], housing_pd_final[18000:19217], hou
sing_pd_final[19215:]
len(train_pd), len(test_pd), len(val_pd)
```

Out[ ]:

```
(18000, 1217, 1218)
```

In [ ]:

```
X_train, y_train = train_pd.to_numpy()[:, :-1], train_pd.to_numpy()[:, -1]
X_val, y_val = val_pd.to_numpy()[:, :-1], val_pd.to_numpy()[:, -1]
X_test, y_test = test_pd.to_numpy()[:, :-1], test_pd.to_numpy()[:, -1]

X_train.shape, y_train.shape, X_val.shape, y_val.shape, X_test.shape, y_test.shape
```

Out[ ]:

```
((18000, 13), (18000,), (1218, 13), (1218,), (1217, 13), (1217,))
```

Feature Scaling & Custom Processor

Scaling ensures all numerical features have a similar range, preventing dominance of features with larger values.

while Preprocessor applies scaling consistently to training, validation, and test sets, ensuring uniform transformation.

In [ ]:

```
scaler = StandardScaler().fit(X_train[:, :8])

def preprocessor(X):
  A = np.copy(X)
  A[:, :8] = scaler.transform(A[:, :8])
  return A

X_train, X_val, X_test = preprocessor(X_train), preprocessor(X_val), preprocessor(X_test
)
```

In [ ]:

```
X_train.shape, X_val.shape, X_test.shape
```

Out[ ]:

```
((18000, 13), (1218, 13), (1217, 13))
```

In [ ]:

```

```

In [ ]:

```
pd.DataFrame(X_train).head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.603443	-0.736073	0.820845	0.081039	0.315396	-0.27684	0.328234	-0.326667	True	False	False	False	False
1	-0.105122	0.537108	0.661774	-0.206526	-0.196843	-0.246809	-0.160526	-0.995001	False	True	False	False	False
2	-1.422454	0.981785	1.377592	-0.374924	-0.301674	0.098553	-0.244613	0.074459	False	False	False	True	False
3	-1.352596	1.019231	1.854804	-1.055419	-1.049782	-1.07974	-1.082862	-1.235508	False	False	False	True	False
4	0.718211	-0.848412	0.661774	-0.507894	-0.589957	-0.628385	-0.567825	0.687448	True	False	False	False	False

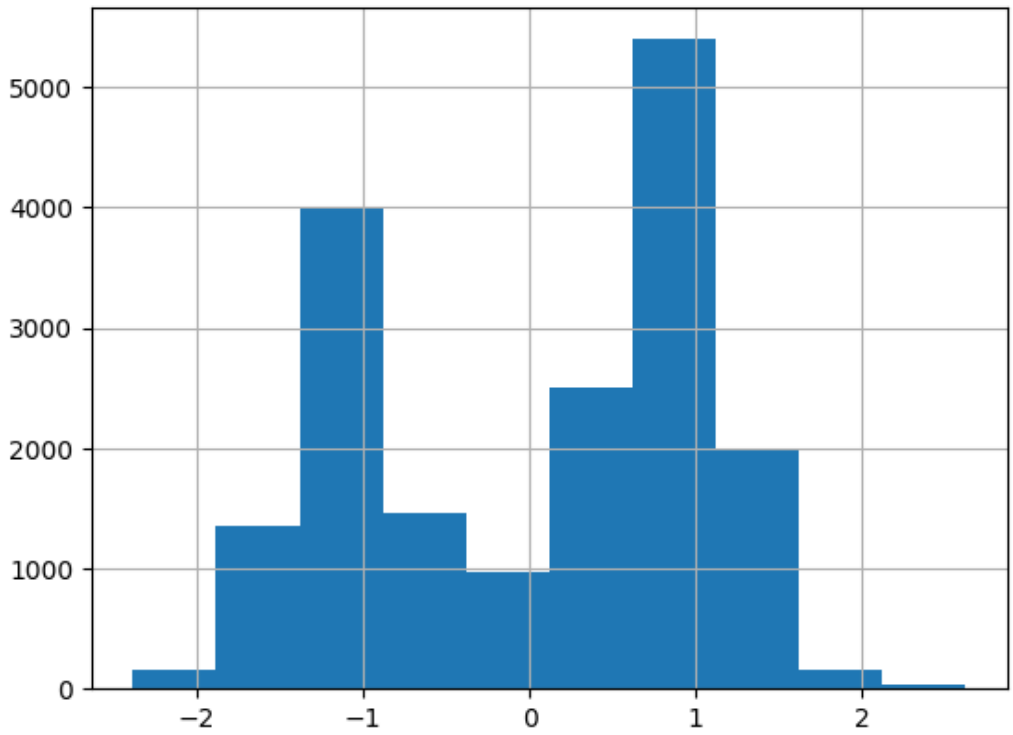
In [ ]:

```
pd.DataFrame(X_train)[0].hist()
```

```
pd.DataFrame(X_train)[0].hist()
```

Out[ ]:

```
<Axes: >
```



A)using Linear Regression

A simple model that predicts house prices by fitting a straight line through the data which May underperform with complex data or non-linear relationships.

In [ ]:

```
lm = LinearRegression().fit(X_train, y_train)
mse(lm.predict(X_train), y_train, squared=False), mse(lm.predict(X_val), y_val, squared=
False)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
(68593.05578127236, 71382.43558330165)
```

B)K-Nearest Neighbors(KNN)

Predicts prices based on the average values of the nearest 10 data points. Limitation: Can overfit if the number of neighbors is too small.

In [ ]:

```
knn = KNeighborsRegressor(n_neighbors=10).fit(X_train, y_train)
mse(knn.predict(X_train), y_train, squared=False), mse(knn.predict(X_val), y_val, square
d=False)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
```

```
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
(53759.09908812057, 62161.22860469906)
```

C)Random Forest

Uses multiple decision trees to make predictions and Handles non-linear data well and reduces overfitting via ensemble learning.

In [ ]:

```
rfr = RandomForestRegressor(max_depth=10).fit(X_train, y_train)
mse(rfr.predict(X_train), y_train, squared=False), mse(rfr.predict(X_val), y_val, square
d=False)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
(43569.9006950773, 53183.737414903306)
```

D)Gradient Boosting

Combines weak learners incrementally to improve accuracy. Typically outperforms Random Forest in predictive tasks.

In [ ]:

```
gbr = GradientBoostingRegressor(n_estimators=250).fit(X_train, y_train)
mse(gbr.predict(X_train), y_train, squared=False), mse(gbr.predict(X_val), y_val, square
d=False)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
(47274.82259072158, 51210.606733472116)
```

In [ ]:

```
# Convert data to float32
X_train = X_train.astype('float32')
X_val = X_val.astype('float32')
y_train = y_train.astype('float32')
y_val = y_val.astype('float32')
```

IMPORTING NECESSARY LIBRARIES AND DEPENDENCIES.

1)for Building Simple Neural Network**

A basic neural network with one hidden layer and two neurons.

In [ ]:

In [ ]:

```
from tensorflow.keras.models import Sequential,load_model
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam

simple_nn = Sequential()
simple_nn.add(InputLayer((13,)))
simple_nn.add(Dense(2, 'relu'))
simple_nn.add(Dense(1, 'linear'))

opt = Adam(learning_rate=.1)
cp = ModelCheckpoint('models/simple_nn.keras', save_best_only=True)  #save as .keras form
at
simple_nn.compile(optimizer=opt, loss='mse', metrics=[RootMeanSquaredError()])
simple_nn.fit(x=X_train, y=y_train, validation_data=(X_val, y_val), callbacks=[cp], epoc
hs=100)
```

```
Epoch 1/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 54539640832.0000 - root_mean_squared_err
or: 233498.9531 - val_loss: 41773555712.0000 - val_root_mean_squared_error: 204385.7969
Epoch 2/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 34940604416.0000 - root_mean_squared_err
or: 186656.0312 - val_loss: 18346248192.0000 - val_root_mean_squared_error: 135448.3281
Epoch 3/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 14725444608.0000 - root_mean_squared_err
or: 121149.6953 - val_loss: 9118205952.0000 - val_root_mean_squared_error: 95489.2969
Epoch 4/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 8249291776.0000 - root_mean_squared_erro
r: 90765.3516 - val_loss: 7016109568.0000 - val_root_mean_squared_error: 83762.2188
Epoch 5/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 6735520256.0000 - root_mean_squared_erro
r: 82020.9297 - val_loss: 6083796480.0000 - val_root_mean_squared_error: 77998.6953
Epoch 6/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5585117696.0000 - root_mean_squared_erro
r: 74721.6016 - val_loss: 5544328704.0000 - val_root_mean_squared_error: 74460.2500
Epoch 7/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 5204871168.0000 - root_mean_squared_erro
r: 72125.4219 - val_loss: 5293306368.0000 - val_root_mean_squared_error: 72755.1094
Epoch 8/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4790865408.0000 - root_mean_squared_erro
r: 69204.6719 - val_loss: 5179879424.0000 - val_root_mean_squared_error: 71971.3828
Epoch 9/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4920813056.0000 - root_mean_squared_erro
r: 70141.4219 - val_loss: 5137507840.0000 - val_root_mean_squared_error: 71676.4141
Epoch 10/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4634716160.0000 - root_mean_squared_erro
r: 68049.6016 - val_loss: 5105841152.0000 - val_root_mean_squared_error: 71455.1719
Epoch 11/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4645094400.0000 - root_mean_squared_erro
r: 68147.4766 - val_loss: 5082308608.0000 - val_root_mean_squared_error: 71290.3125
Epoch 12/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4717749248.0000 - root_mean_squared_erro
r: 68681.7344 - val_loss: 5066271744.0000 - val_root_mean_squared_error: 71177.7500
Epoch 13/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4613076992.0000 - root_mean_squared_erro
r: 67911.8516 - val_loss: 5059229184.0000 - val_root_mean_squared_error: 71128.2578
Epoch 14/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4455755776.0000 - root_mean_squared_erro
r: 66721.1484 - val_loss: 5058302464.0000 - val_root_mean_squared_error: 71121.7422
Epoch 15/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4638859776.0000 - root_mean_squared_erro
r: 68104.7422 - val_loss: 5044654592.0000 - val_root_mean_squared_error: 71025.7344
Epoch 16/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4555708928.0000 - root_mean_squared_erro
r: 67487.5391 - val_loss: 5044612096.0000 - val_root_mean_squared_error: 71025.4297
Epoch 17/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4690583552.0000 - root_mean_squared_erro
r: 68484.6562 - val_loss: 5053232640.0000 - val_root_mean_squared_error: 71086.0938
Epoch 18/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4631273472.0000 - root_mean_squared_erro
```

r: 68047.0234 - val_loss: 5029160448.0000 - val_root_mean_squared_error: 70916.5703
Epoch 19/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4499521536.0000 - root_mean_squared_erro
r: 67056.4922 - val_loss: 5035070976.0000 - val_root_mean_squared_error: 70958.2344
Epoch 20/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4797551616.0000 - root_mean_squared_erro
r: 69251.0391 - val_loss: 5032795648.0000 - val_root_mean_squared_error: 70942.2031
Epoch 21/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4562572288.0000 - root_mean_squared_erro
r: 67525.8828 - val_loss: 5033680896.0000 - val_root_mean_squared_error: 70948.4375
Epoch 22/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4501138944.0000 - root_mean_squared_erro
r: 67079.5312 - val_loss: 5014000128.0000 - val_root_mean_squared_error: 70809.6016
Epoch 23/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4566908416.0000 - root_mean_squared_erro
r: 67565.7188 - val_loss: 5000335872.0000 - val_root_mean_squared_error: 70713.0547
Epoch 24/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4560048640.0000 - root_mean_squared_erro
r: 67522.9766 - val_loss: 4997998080.0000 - val_root_mean_squared_error: 70696.5234
Epoch 25/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4510138368.0000 - root_mean_squared_erro
r: 67145.5312 - val_loss: 5007716864.0000 - val_root_mean_squared_error: 70765.2266
Epoch 26/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4602163712.0000 - root_mean_squared_erro
r: 67835.9141 - val_loss: 5010356224.0000 - val_root_mean_squared_error: 70783.8672
Epoch 27/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4547445248.0000 - root_mean_squared_erro
r: 67426.8125 - val_loss: 4978567680.0000 - val_root_mean_squared_error: 70558.9688
Epoch 28/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4559151104.0000 - root_mean_squared_erro
r: 67515.2031 - val_loss: 4975286784.0000 - val_root_mean_squared_error: 70535.7109
Epoch 29/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4496536576.0000 - root_mean_squared_erro
r: 67046.3047 - val_loss: 4982700032.0000 - val_root_mean_squared_error: 70588.2422
Epoch 30/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4587983872.0000 - root_mean_squared_erro
r: 67727.4297 - val_loss: 4969357312.0000 - val_root_mean_squared_error: 70493.6719
Epoch 31/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4791488000.0000 - root_mean_squared_erro
r: 69181.0938 - val_loss: 4949418496.0000 - val_root_mean_squared_error: 70352.1016
Epoch 32/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4594275328.0000 - root_mean_squared_erro
r: 67778.4844 - val_loss: 4962390528.0000 - val_root_mean_squared_error: 70444.2344
Epoch 33/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4399966208.0000 - root_mean_squared_erro
r: 66321.8594 - val_loss: 4965132800.0000 - val_root_mean_squared_error: 70463.6953
Epoch 34/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4614180864.0000 - root_mean_squared_erro
r: 67918.1641 - val_loss: 4946462208.0000 - val_root_mean_squared_error: 70331.0938
Epoch 35/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4390273536.0000 - root_mean_squared_erro
r: 66254.6250 - val_loss: 4944105984.0000 - val_root_mean_squared_error: 70314.3359
Epoch 36/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4382669312.0000 - root_mean_squared_erro
r: 66193.2578 - val_loss: 4909303296.0000 - val_root_mean_squared_error: 70066.4219
Epoch 37/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 4607254016.0000 - root_mean_squared_erro
r: 67868.2578 - val_loss: 4921146880.0000 - val_root_mean_squared_error: 70150.8828
Epoch 38/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4531285504.0000 - root_mean_squared_erro
r: 67308.3594 - val_loss: 4902928384.0000 - val_root_mean_squared_error: 70020.9141
Epoch 39/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - loss: 4444804096.0000 - root_mean_squared_erro
r: 66641.3906 - val_loss: 4904748544.0000 - val_root_mean_squared_error: 70033.9062
Epoch 40/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4475655168.0000 - root_mean_squared_erro
r: 66892.1406 - val_loss: 4873295360.0000 - val_root_mean_squared_error: 69808.9922
Epoch 41/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4526555648.0000 - root_mean_squared_erro
r: 67273.4688 - val_loss: 4886693376.0000 - val_root_mean_squared_error: 69904.8906
Epoch 42/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4438777344.0000 - root_mean_squared_erro

r: 66620.0234 - val_loss: 4863424000.0000 - val_root_mean_squared_error: 69738.2500
Epoch 43/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4529377280.0000 - root_mean_squared_erro
r: 67296.2734 - val_loss: 4862021120.0000 - val_root_mean_squared_error: 69728.1953
Epoch 44/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4472200192.0000 - root_mean_squared_erro
r: 66858.0000 - val_loss: 4856575488.0000 - val_root_mean_squared_error: 69689.1328
Epoch 45/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4572540416.0000 - root_mean_squared_erro
r: 67595.7578 - val_loss: 4859686400.0000 - val_root_mean_squared_error: 69711.4531
Epoch 46/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4486858752.0000 - root_mean_squared_erro
r: 66978.5234 - val_loss: 4835853312.0000 - val_root_mean_squared_error: 69540.2969
Epoch 47/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4441146368.0000 - root_mean_squared_erro
r: 66638.9844 - val_loss: 4842146304.0000 - val_root_mean_squared_error: 69585.5312
Epoch 48/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4439943680.0000 - root_mean_squared_erro
r: 66622.0234 - val_loss: 4845543936.0000 - val_root_mean_squared_error: 69609.9375
Epoch 49/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4431044608.0000 - root_mean_squared_erro
r: 66549.9219 - val_loss: 4802679296.0000 - val_root_mean_squared_error: 69301.3672
Epoch 50/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4457390592.0000 - root_mean_squared_erro
r: 66756.8906 - val_loss: 4829318656.0000 - val_root_mean_squared_error: 69493.2969
Epoch 51/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4307536896.0000 - root_mean_squared_erro
r: 65626.4453 - val_loss: 4807420416.0000 - val_root_mean_squared_error: 69335.5625
Epoch 52/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4259789056.0000 - root_mean_squared_erro
r: 65260.5898 - val_loss: 4793296384.0000 - val_root_mean_squared_error: 69233.6328
Epoch 53/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4422423040.0000 - root_mean_squared_erro
r: 66490.9844 - val_loss: 4811170816.0000 - val_root_mean_squared_error: 69362.6016
Epoch 54/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4500658688.0000 - root_mean_squared_erro
r: 67070.8438 - val_loss: 4789377536.0000 - val_root_mean_squared_error: 69205.3281
Epoch 55/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4304159232.0000 - root_mean_squared_erro
r: 65601.5312 - val_loss: 4777832448.0000 - val_root_mean_squared_error: 69121.8672
Epoch 56/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 4398196224.0000 - root_mean_squared_erro
r: 66304.4375 - val_loss: 4777438720.0000 - val_root_mean_squared_error: 69119.0156
Epoch 57/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 4544287232.0000 - root_mean_squared_erro
r: 67397.7500 - val_loss: 4761050624.0000 - val_root_mean_squared_error: 69000.3672
Epoch 58/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4450718208.0000 - root_mean_squared_erro
r: 66707.0781 - val_loss: 4784960000.0000 - val_root_mean_squared_error: 69173.4062
Epoch 59/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4431842304.0000 - root_mean_squared_erro
r: 66542.6797 - val_loss: 4785341440.0000 - val_root_mean_squared_error: 69176.1641
Epoch 60/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4287498752.0000 - root_mean_squared_erro
r: 65451.5156 - val_loss: 4753483264.0000 - val_root_mean_squared_error: 68945.5078
Epoch 61/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4450786816.0000 - root_mean_squared_erro
r: 66710.0938 - val_loss: 4743922688.0000 - val_root_mean_squared_error: 68876.1406
Epoch 62/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4459515904.0000 - root_mean_squared_erro
r: 66775.8594 - val_loss: 4735197696.0000 - val_root_mean_squared_error: 68812.7734
Epoch 63/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4416309760.0000 - root_mean_squared_erro
r: 66443.1641 - val_loss: 4752134144.0000 - val_root_mean_squared_error: 68935.7266
Epoch 64/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4311967744.0000 - root_mean_squared_erro
r: 65660.7891 - val_loss: 4742831104.0000 - val_root_mean_squared_error: 68868.2188
Epoch 65/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4292076800.0000 - root_mean_squared_erro
r: 65511.1094 - val_loss: 4729505792.0000 - val_root_mean_squared_error: 68771.4062
Epoch 66/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4455957504.0000 - root_mean_squared_erro

r: 66730.6172 - val_loss: 4739006976.0000 - val_root_mean_squared_error: 68840.4453
Epoch 67/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 4397680128.0000 - root_mean_squared_erro
r: 66299.3281 - val_loss: 4713782784.0000 - val_root_mean_squared_error: 68656.9922
Epoch 68/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4305144832.0000 - root_mean_squared_erro
r: 65596.6016 - val_loss: 4700944896.0000 - val_root_mean_squared_error: 68563.4375
Epoch 69/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4272348672.0000 - root_mean_squared_erro
r: 65358.3047 - val_loss: 4702661120.0000 - val_root_mean_squared_error: 68575.9531
Epoch 70/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4496883712.0000 - root_mean_squared_erro
r: 67040.0156 - val_loss: 4701617152.0000 - val_root_mean_squared_error: 68568.3359
Epoch 71/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4281522432.0000 - root_mean_squared_erro
r: 65429.9180 - val_loss: 4705542656.0000 - val_root_mean_squared_error: 68596.9609
Epoch 72/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4392652288.0000 - root_mean_squared_erro
r: 66248.1250 - val_loss: 4678725632.0000 - val_root_mean_squared_error: 68401.2109
Epoch 73/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4228637440.0000 - root_mean_squared_erro
r: 65016.9023 - val_loss: 4679358464.0000 - val_root_mean_squared_error: 68405.8359
Epoch 74/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4343191040.0000 - root_mean_squared_erro
r: 65896.5000 - val_loss: 4663543808.0000 - val_root_mean_squared_error: 68290.1406
Epoch 75/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4283275008.0000 - root_mean_squared_erro
r: 65443.3281 - val_loss: 4677759488.0000 - val_root_mean_squared_error: 68394.1484
Epoch 76/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4281000448.0000 - root_mean_squared_erro
r: 65425.3320 - val_loss: 4647067136.0000 - val_root_mean_squared_error: 68169.3984
Epoch 77/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4335552512.0000 - root_mean_squared_erro
r: 65823.2734 - val_loss: 4649236480.0000 - val_root_mean_squared_error: 68185.3125
Epoch 78/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4189521152.0000 - root_mean_squared_erro
r: 64711.1641 - val_loss: 4629218816.0000 - val_root_mean_squared_error: 68038.3594
Epoch 79/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4243095040.0000 - root_mean_squared_erro
r: 65132.8594 - val_loss: 4641016320.0000 - val_root_mean_squared_error: 68125.0078
Epoch 80/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4345121792.0000 - root_mean_squared_erro
r: 65905.9609 - val_loss: 4659392512.0000 - val_root_mean_squared_error: 68259.7422
Epoch 81/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4396019200.0000 - root_mean_squared_erro
r: 66294.7422 - val_loss: 4643544064.0000 - val_root_mean_squared_error: 68143.5547
Epoch 82/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4285911040.0000 - root_mean_squared_erro
r: 65454.3555 - val_loss: 4642816000.0000 - val_root_mean_squared_error: 68138.2109
Epoch 83/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4263274496.0000 - root_mean_squared_erro
r: 65276.7070 - val_loss: 4611879424.0000 - val_root_mean_squared_error: 67910.8203
Epoch 84/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4195711232.0000 - root_mean_squared_erro
r: 64763.8086 - val_loss: 4605697024.0000 - val_root_mean_squared_error: 67865.2891
Epoch 85/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4130084864.0000 - root_mean_squared_erro
r: 64248.7773 - val_loss: 4629566976.0000 - val_root_mean_squared_error: 68040.9219
Epoch 86/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 4174304256.0000 - root_mean_squared_erro
r: 64597.7461 - val_loss: 4603520000.0000 - val_root_mean_squared_error: 67849.2422
Epoch 87/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4204616704.0000 - root_mean_squared_erro
r: 64834.9023 - val_loss: 4597460992.0000 - val_root_mean_squared_error: 67804.5781
Epoch 88/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4314948608.0000 - root_mean_squared_erro
r: 65679.0078 - val_loss: 4628914688.0000 - val_root_mean_squared_error: 68036.1250
Epoch 89/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4340725248.0000 - root_mean_squared_erro
r: 65879.4609 - val_loss: 4597753856.0000 - val_root_mean_squared_error: 67806.7422
Epoch 90/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4320001024.0000 - root_mean_squared_erro

```
r: 65714.3438 - val_loss: 4596164608.0000 - val_root_mean_squared_error: 67795.0156
Epoch 91/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4308797952.0000 - root_mean_squared_erro
r: 65628.4062 - val_loss: 4600026112.0000 - val_root_mean_squared_error: 67823.4922
Epoch 92/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4294361600.0000 - root_mean_squared_erro
r: 65529.2266 - val_loss: 4600787968.0000 - val_root_mean_squared_error: 67829.1094
Epoch 93/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4311780352.0000 - root_mean_squared_erro
r: 65659.9688 - val_loss: 4594822656.0000 - val_root_mean_squared_error: 67785.1250
Epoch 94/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4289721856.0000 - root_mean_squared_erro
r: 65487.6602 - val_loss: 4602252800.0000 - val_root_mean_squared_error: 67839.9062
Epoch 95/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 4193928960.0000 - root_mean_squared_erro
r: 64746.9375 - val_loss: 4600219136.0000 - val_root_mean_squared_error: 67824.9141
Epoch 96/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4257855232.0000 - root_mean_squared_erro
r: 65236.8750 - val_loss: 4589726208.0000 - val_root_mean_squared_error: 67747.5156
Epoch 97/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4367850496.0000 - root_mean_squared_erro
r: 66035.8672 - val_loss: 4570818048.0000 - val_root_mean_squared_error: 67607.8281
Epoch 98/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4291658752.0000 - root_mean_squared_erro
r: 65482.3906 - val_loss: 4583065088.0000 - val_root_mean_squared_error: 67698.3359
Epoch 99/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4103278848.0000 - root_mean_squared_erro
r: 64045.1992 - val_loss: 4571387904.0000 - val_root_mean_squared_error: 67612.0391
Epoch 100/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4221046016.0000 - root_mean_squared_erro
r: 64949.8672 - val_loss: 4567960576.0000 - val_root_mean_squared_error: 67586.6875
```

Out[ ]:

```
<keras.src.callbacks.history.History at 0x7880582c0df0>
```

In [ ]:

```
simple_nn = load_model('models/simple_nn.keras')
mse(simple_nn.predict(X_train), y_train, squared=False), mse(simple_nn.predict(X_val), y
_val, squared=False)
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
39/39 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
(65134.406, 67586.695)
```

2)for Medium Neural Network

Adds more complexity with two hidden layers (32 and 16 neurons)

In [ ]:

```
medium_nn = Sequential()
medium_nn.add(InputLayer((13,)))
medium_nn.add(Dense(32, 'relu'))
medium_nn.add(Dense(16, 'relu'))
medium_nn.add(Dense(1, 'linear'))

opt = Adam(learning_rate=.1)
cp = ModelCheckpoint('models/medium_nn.keras', save_best_only=True)
```

```
medium_nn.compile(optimizer=opt, loss='mse', metrics=[RootMeanSquaredError()])
medium_nn.fit(x=X_train, y=y_train, validation_data=(X_val, y_val), callbacks=[cp], epochs=100)
```

```
Epoch 1/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 19243558912.0000 - root_mean_squared_error: 132632.7500 - val_loss: 5051724288.0000 - val_root_mean_squared_error: 71075.4844
Epoch 2/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 4470901760.0000 - root_mean_squared_error: 66862.2266 - val_loss: 4687271424.0000 - val_root_mean_squared_error: 68463.6484
Epoch 3/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4235913472.0000 - root_mean_squared_error: 65050.3984 - val_loss: 4779671552.0000 - val_root_mean_squared_error: 69135.1719
Epoch 4/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 4214558720.0000 - root_mean_squared_error: 64906.3906 - val_loss: 4535901696.0000 - val_root_mean_squared_error: 67349.1016
Epoch 5/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4208945664.0000 - root_mean_squared_error: 64865.4570 - val_loss: 4441363968.0000 - val_root_mean_squared_error: 66643.5625
Epoch 6/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 4287689472.0000 - root_mean_squared_error: 65476.4062 - val_loss: 4505629696.0000 - val_root_mean_squared_error: 67123.9844
Epoch 7/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4231639808.0000 - root_mean_squared_error: 65039.0117 - val_loss: 4413633536.0000 - val_root_mean_squared_error: 66435.1797
Epoch 8/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 4149835520.0000 - root_mean_squared_error: 64411.9062 - val_loss: 4415021056.0000 - val_root_mean_squared_error: 66445.6250
Epoch 9/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 4095428352.0000 - root_mean_squared_error: 63988.2031 - val_loss: 4243283968.0000 - val_root_mean_squared_error: 65140.4922
Epoch 10/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4011297536.0000 - root_mean_squared_error: 63327.5742 - val_loss: 4294592256.0000 - val_root_mean_squared_error: 65533.1367
Epoch 11/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4146739200.0000 - root_mean_squared_error: 64383.9141 - val_loss: 4316926464.0000 - val_root_mean_squared_error: 65703.3203
Epoch 12/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3931311360.0000 - root_mean_squared_error: 62678.6016 - val_loss: 4340554752.0000 - val_root_mean_squared_error: 65882.8828
Epoch 13/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4063308800.0000 - root_mean_squared_error: 63733.9688 - val_loss: 4332282368.0000 - val_root_mean_squared_error: 65820.0781
Epoch 14/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3925444096.0000 - root_mean_squared_error: 62646.2617 - val_loss: 4260986624.0000 - val_root_mean_squared_error: 65276.2344
Epoch 15/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3991703040.0000 - root_mean_squared_error: 63172.4375 - val_loss: 4311772672.0000 - val_root_mean_squared_error: 65664.0859
Epoch 16/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4127987712.0000 - root_mean_squared_error: 64226.8750 - val_loss: 4260505600.0000 - val_root_mean_squared_error: 65272.5469
Epoch 17/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3937495808.0000 - root_mean_squared_error: 62745.0742 - val_loss: 4289468416.0000 - val_root_mean_squared_error: 65494.0352
Epoch 18/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 3990547712.0000 - root_mean_squared_error: 63160.7188 - val_loss: 4455271936.0000 - val_root_mean_squared_error: 66747.8203
Epoch 19/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4035778816.0000 - root_mean_squared_error: 63519.3789 - val_loss: 4305367040.0000 - val_root_mean_squared_error: 65615.2969
Epoch 20/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4104944896.0000 - root_mean_squared_error: 64061.6406 - val_loss: 4284664064.0000 - val_root_mean_squared_error: 65457.3438
Epoch 21/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3989139968.0000 - root_mean_squared_error: 63148.9648 - val_loss: 4303666688.0000 - val_root_mean_squared_error: 65602.3359
Epoch 22/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4117455104.0000 - root_mean_squared_error: 64154.7969 - val_loss: 4634318336.0000 - val_root_mean_squared_error: 68075.8281
Epoch 23/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3940097536.0000 - root_mean_squared_error:
```

```
r: 62767.7305 - val_loss: 4259236608.0000 - val_root_mean_squared_error: 65262.8281
Epoch 24/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3827083264.0000 - root_mean_squared_erro
r: 61811.9609 - val_loss: 4191153152.0000 - val_root_mean_squared_error: 64739.1172
Epoch 25/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4119893760.0000 - root_mean_squared_erro
r: 64167.0195 - val_loss: 4347846144.0000 - val_root_mean_squared_error: 65938.2031
Epoch 26/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3881918464.0000 - root_mean_squared_erro
r: 62282.4531 - val_loss: 4245288192.0000 - val_root_mean_squared_error: 65155.8750
Epoch 27/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3891435264.0000 - root_mean_squared_erro
r: 62360.9375 - val_loss: 4392724992.0000 - val_root_mean_squared_error: 66277.6328
Epoch 28/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 3850518016.0000 - root_mean_squared_erro
r: 62037.5703 - val_loss: 4091012352.0000 - val_root_mean_squared_error: 63961.0234
Epoch 29/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3861900288.0000 - root_mean_squared_erro
r: 62134.2930 - val_loss: 4082724864.0000 - val_root_mean_squared_error: 63896.2031
Epoch 30/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3839528448.0000 - root_mean_squared_erro
r: 61948.6484 - val_loss: 4060268032.0000 - val_root_mean_squared_error: 63720.2344
Epoch 31/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3783179008.0000 - root_mean_squared_erro
r: 61499.1680 - val_loss: 3976538624.0000 - val_root_mean_squared_error: 63059.8008
Epoch 32/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3713291264.0000 - root_mean_squared_erro
r: 60918.9844 - val_loss: 4084550144.0000 - val_root_mean_squared_error: 63910.4844
Epoch 33/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3541713664.0000 - root_mean_squared_erro
r: 59499.5117 - val_loss: 3814196992.0000 - val_root_mean_squared_error: 61759.1836
Epoch 34/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3473965568.0000 - root_mean_squared_erro
r: 58936.3125 - val_loss: 3815220480.0000 - val_root_mean_squared_error: 61767.4727
Epoch 35/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3447288576.0000 - root_mean_squared_erro
r: 58709.3945 - val_loss: 3837622016.0000 - val_root_mean_squared_error: 61948.5430
Epoch 36/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3396995584.0000 - root_mean_squared_erro
r: 58267.9570 - val_loss: 3692597504.0000 - val_root_mean_squared_error: 60766.7461
Epoch 37/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3329153024.0000 - root_mean_squared_erro
r: 57675.5938 - val_loss: 3614981376.0000 - val_root_mean_squared_error: 60124.7148
Epoch 38/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 3297738496.0000 - root_mean_squared_erro
r: 57417.2227 - val_loss: 3780383744.0000 - val_root_mean_squared_error: 61484.8242
Epoch 39/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3337052672.0000 - root_mean_squared_erro
r: 57760.2891 - val_loss: 3676929792.0000 - val_root_mean_squared_error: 60637.6914
Epoch 40/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3301145600.0000 - root_mean_squared_erro
r: 57443.7188 - val_loss: 3597488384.0000 - val_root_mean_squared_error: 59979.0664
Epoch 41/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3162695424.0000 - root_mean_squared_erro
r: 56233.1133 - val_loss: 3566669824.0000 - val_root_mean_squared_error: 59721.6016
Epoch 42/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3177044992.0000 - root_mean_squared_erro
r: 56362.4102 - val_loss: 3536328704.0000 - val_root_mean_squared_error: 59467.0391
Epoch 43/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3268365824.0000 - root_mean_squared_erro
r: 57143.3867 - val_loss: 3583668992.0000 - val_root_mean_squared_error: 59863.7539
Epoch 44/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3094115840.0000 - root_mean_squared_erro
r: 55611.1758 - val_loss: 3460246528.0000 - val_root_mean_squared_error: 58823.8594
Epoch 45/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3105144576.0000 - root_mean_squared_erro
r: 55711.4805 - val_loss: 3523337216.0000 - val_root_mean_squared_error: 59357.7070
Epoch 46/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3067174144.0000 - root_mean_squared_erro
r: 55361.6055 - val_loss: 3438238720.0000 - val_root_mean_squared_error: 58636.4961
Epoch 47/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 3078836224.0000 - root_mean_squared_erro
```

r: 55476.2773 - val_loss: 3628026368.0000 - val_root_mean_squared_error: 60233.1016
Epoch 48/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3000899584.0000 - root_mean_squared_erro
r: 54762.2031 - val_loss: 3434891264.0000 - val_root_mean_squared_error: 58607.9453
Epoch 49/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3115786496.0000 - root_mean_squared_erro
r: 55798.1094 - val_loss: 3468494592.0000 - val_root_mean_squared_error: 58893.9258
Epoch 50/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3023492864.0000 - root_mean_squared_erro
r: 54976.3477 - val_loss: 3419905536.0000 - val_root_mean_squared_error: 58479.9570
Epoch 51/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3084845056.0000 - root_mean_squared_erro
r: 55530.3164 - val_loss: 3402247680.0000 - val_root_mean_squared_error: 58328.7891
Epoch 52/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3087880448.0000 - root_mean_squared_erro
r: 55561.1797 - val_loss: 3348767744.0000 - val_root_mean_squared_error: 57868.5391
Epoch 53/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2907374592.0000 - root_mean_squared_erro
r: 53887.1172 - val_loss: 3699054080.0000 - val_root_mean_squared_error: 60819.8477
Epoch 54/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3145741312.0000 - root_mean_squared_erro
r: 56050.2734 - val_loss: 3385556992.0000 - val_root_mean_squared_error: 58185.5391
Epoch 55/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2900940544.0000 - root_mean_squared_erro
r: 53845.1445 - val_loss: 3328689152.0000 - val_root_mean_squared_error: 57694.7930
Epoch 56/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3002795008.0000 - root_mean_squared_erro
r: 54788.5547 - val_loss: 3369355776.0000 - val_root_mean_squared_error: 58046.1523
Epoch 57/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2948444416.0000 - root_mean_squared_erro
r: 54295.2383 - val_loss: 3412904960.0000 - val_root_mean_squared_error: 58420.0742
Epoch 58/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2999850752.0000 - root_mean_squared_erro
r: 54763.9961 - val_loss: 3373189632.0000 - val_root_mean_squared_error: 58079.1680
Epoch 59/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2920975616.0000 - root_mean_squared_erro
r: 54038.0391 - val_loss: 3357112832.0000 - val_root_mean_squared_error: 57940.5977
Epoch 60/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2934187264.0000 - root_mean_squared_erro
r: 54162.1406 - val_loss: 3397948672.0000 - val_root_mean_squared_error: 58291.9258
Epoch 61/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3040012288.0000 - root_mean_squared_erro
r: 55123.9609 - val_loss: 3299003904.0000 - val_root_mean_squared_error: 57436.9570
Epoch 62/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2953530880.0000 - root_mean_squared_erro
r: 54339.9414 - val_loss: 3324859648.0000 - val_root_mean_squared_error: 57661.5977
Epoch 63/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2907568896.0000 - root_mean_squared_erro
r: 53917.2422 - val_loss: 3333906176.0000 - val_root_mean_squared_error: 57739.9883
Epoch 64/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2916231168.0000 - root_mean_squared_erro
r: 53995.5977 - val_loss: 3450786048.0000 - val_root_mean_squared_error: 58743.3906
Epoch 65/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2947845120.0000 - root_mean_squared_erro
r: 54275.5703 - val_loss: 3378517248.0000 - val_root_mean_squared_error: 58125.0156
Epoch 66/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2827737344.0000 - root_mean_squared_erro
r: 53165.6797 - val_loss: 3598701312.0000 - val_root_mean_squared_error: 59989.1758
Epoch 67/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2910422784.0000 - root_mean_squared_erro
r: 53943.3750 - val_loss: 3412234240.0000 - val_root_mean_squared_error: 58414.3320
Epoch 68/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2998081792.0000 - root_mean_squared_erro
r: 54745.0117 - val_loss: 3675497728.0000 - val_root_mean_squared_error: 60625.8828
Epoch 69/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2865273856.0000 - root_mean_squared_erro
r: 53506.3477 - val_loss: 3298195456.0000 - val_root_mean_squared_error: 57429.9180
Epoch 70/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3017387776.0000 - root_mean_squared_erro
r: 54894.9062 - val_loss: 3316144384.0000 - val_root_mean_squared_error: 57585.9727
Epoch 71/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2856635648.0000 - root_mean_squared_erro

```
r: 53440.0859 - val_loss: 3250820864.0000 - val_root_mean_squared_error: 57015.9688
Epoch 72/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2940389888.0000 - root_mean_squared_erro
r: 54184.2344 - val_loss: 3554553856.0000 - val_root_mean_squared_error: 59620.0781
Epoch 73/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2920979456.0000 - root_mean_squared_erro
r: 54036.2305 - val_loss: 3320546304.0000 - val_root_mean_squared_error: 57624.1797
Epoch 74/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2898277376.0000 - root_mean_squared_erro
r: 53821.5703 - val_loss: 3397987840.0000 - val_root_mean_squared_error: 58292.2617
Epoch 75/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2936926464.0000 - root_mean_squared_erro
r: 54170.9727 - val_loss: 3197381632.0000 - val_root_mean_squared_error: 56545.3945
Epoch 76/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2835487232.0000 - root_mean_squared_erro
r: 53238.1719 - val_loss: 3338370048.0000 - val_root_mean_squared_error: 57778.6289
Epoch 77/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2803768064.0000 - root_mean_squared_erro
r: 52944.4375 - val_loss: 3267643136.0000 - val_root_mean_squared_error: 57163.3008
Epoch 78/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2767419392.0000 - root_mean_squared_erro
r: 52602.3125 - val_loss: 3323363584.0000 - val_root_mean_squared_error: 57648.6211
Epoch 79/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2731195904.0000 - root_mean_squared_erro
r: 52252.7969 - val_loss: 3232067584.0000 - val_root_mean_squared_error: 56851.2773
Epoch 80/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2695593984.0000 - root_mean_squared_erro
r: 51901.3164 - val_loss: 3266189824.0000 - val_root_mean_squared_error: 57150.5898
Epoch 81/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2820049152.0000 - root_mean_squared_erro
r: 53094.2461 - val_loss: 3329429504.0000 - val_root_mean_squared_error: 57701.2070
Epoch 82/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2812450560.0000 - root_mean_squared_erro
r: 53017.9141 - val_loss: 3207268608.0000 - val_root_mean_squared_error: 56632.7539
Epoch 83/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2801030912.0000 - root_mean_squared_erro
r: 52918.9570 - val_loss: 3256188672.0000 - val_root_mean_squared_error: 57063.0234
Epoch 84/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2736392448.0000 - root_mean_squared_erro
r: 52306.1055 - val_loss: 3294946816.0000 - val_root_mean_squared_error: 57401.6289
Epoch 85/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2802924800.0000 - root_mean_squared_erro
r: 52934.9922 - val_loss: 3177144832.0000 - val_root_mean_squared_error: 56366.1680
Epoch 86/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2694107136.0000 - root_mean_squared_erro
r: 51880.6758 - val_loss: 3183791616.0000 - val_root_mean_squared_error: 56425.0977
Epoch 87/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2705714176.0000 - root_mean_squared_erro
r: 52010.1016 - val_loss: 3172370432.0000 - val_root_mean_squared_error: 56323.8008
Epoch 88/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2757981952.0000 - root_mean_squared_erro
r: 52504.8828 - val_loss: 3556987904.0000 - val_root_mean_squared_error: 59640.4883
Epoch 89/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2719256576.0000 - root_mean_squared_erro
r: 52134.2852 - val_loss: 3156543488.0000 - val_root_mean_squared_error: 56183.1250
Epoch 90/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2702067968.0000 - root_mean_squared_erro
r: 51965.3047 - val_loss: 3149921536.0000 - val_root_mean_squared_error: 56124.1602
Epoch 91/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2615707648.0000 - root_mean_squared_erro
r: 51122.2734 - val_loss: 3196517120.0000 - val_root_mean_squared_error: 56537.7500
Epoch 92/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2786473216.0000 - root_mean_squared_erro
r: 52777.5391 - val_loss: 3376427776.0000 - val_root_mean_squared_error: 58107.0391
Epoch 93/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2750332672.0000 - root_mean_squared_erro
r: 52439.9453 - val_loss: 3351586560.0000 - val_root_mean_squared_error: 57892.8906
Epoch 94/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2717035776.0000 - root_mean_squared_erro
r: 52118.0703 - val_loss: 3342032640.0000 - val_root_mean_squared_error: 57810.3164
Epoch 95/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2710880768.0000 - root_mean_squared_erro
```

```
r: 52056.3086 - val_loss: 3274188800.0000 - val_root_mean_squared_error: 57220.5273
Epoch 96/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2720423680.0000 - root_mean_squared_erro
r: 52140.6211 - val_loss: 3125320960.0000 - val_root_mean_squared_error: 55904.5703
Epoch 97/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2712364288.0000 - root_mean_squared_erro
r: 52068.3398 - val_loss: 3376818176.0000 - val_root_mean_squared_error: 58110.3945
Epoch 98/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2600215040.0000 - root_mean_squared_erro
r: 50972.1562 - val_loss: 3128856320.0000 - val_root_mean_squared_error: 55936.1797
Epoch 99/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2713232128.0000 - root_mean_squared_erro
r: 52086.4727 - val_loss: 3166954240.0000 - val_root_mean_squared_error: 56275.6992
Epoch 100/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2770647808.0000 - root_mean_squared_erro
r: 52631.8164 - val_loss: 3049800192.0000 - val_root_mean_squared_error: 55224.9961
```

Out[ ]:

```
<keras.src.callbacks.history.History at 0x7880608257b0>
```

In [ ]:

```
medium_nn = load_model('models/medium_nn.keras')
mse(medium_nn.predict(X_train), y_train, squared=False), mse(medium_nn.predict(X_val), y
_val, squared=False)
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
39/39 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
(52156.613, 55225.0)
```

3) for Large Neural Network

A highly complex network with multiple layers, useful for capturing intricate patterns.

In [ ]:

```
large_nn = Sequential()
large_nn.add(InputLayer((13,)))
large_nn.add(Dense(256, 'relu'))
large_nn.add(Dense(128, 'relu'))
large_nn.add(Dense(64, 'relu'))
large_nn.add(Dense(32, 'relu'))
large_nn.add(Dense(1, 'linear'))

#Training Neural Network
#Neural networks require iterative optimization, where the optimizer (e.g., Adam) adjusts
weights to minimize error.
opt = Adam(learning_rate=.1)
cp = ModelCheckpoint('models/large_nn.keras', save_best_only=True)
large_nn.compile(optimizer=opt, loss='mse', metrics=[RootMeanSquaredError()])
large_nn.fit(x=X_train, y=y_train, validation_data=(X_val, y_val), callbacks=[cp], epoch
s=100)
```

```
Epoch 1/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 14577685504.0000 - root_mean_squared_err
or: 115639.1953 - val_loss: 6201869312.0000 - val_root_mean_squared_error: 78751.9453
Epoch 2/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 4583801344.0000 - root_mean_squared_erro
r: 67686.6641 - val_loss: 4813209600.0000 - val_root_mean_squared_error: 69377.2969
Epoch 3/100
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 4220409344.0000 - root_mean_squared_erro
r: 64951.8555 - val_loss: 5594952704.0000 - val_root_mean_squared_error: 74799.4141
Epoch 4/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 4146496256.0000 - root_mean_squared_erro
r: 64378.2812 - val_loss: 4005124608.0000 - val_root_mean_squared_error: 63286.0547
Epoch 5/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3633795328.0000 - root_mean_squared_erro
r: 60272.2188 - val_loss: 4249569536.0000 - val_root_mean_squared_error: 65188.7227
Epoch 6/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3642889728.0000 - root_mean_squared_erro
r: 60349.5273 - val_loss: 3903959808.0000 - val_root_mean_squared_error: 62481.6758
Epoch 7/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3608773120.0000 - root_mean_squared_erro
r: 60055.6719 - val_loss: 3891719680.0000 - val_root_mean_squared_error: 62383.6484
Epoch 8/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 3386950144.0000 - root_mean_squared_erro
r: 58176.4805 - val_loss: 3559396864.0000 - val_root_mean_squared_error: 59660.6797
Epoch 9/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3472865280.0000 - root_mean_squared_erro
r: 58906.1953 - val_loss: 3635167232.0000 - val_root_mean_squared_error: 60292.3477
Epoch 10/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 3272971520.0000 - root_mean_squared_erro
r: 57186.0938 - val_loss: 4142724096.0000 - val_root_mean_squared_error: 64363.9961
Epoch 11/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 3476674304.0000 - root_mean_squared_erro
r: 58941.3984 - val_loss: 4091129856.0000 - val_root_mean_squared_error: 63961.9414
Epoch 12/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 3373266432.0000 - root_mean_squared_erro
r: 58050.8008 - val_loss: 3671097344.0000 - val_root_mean_squared_error: 60589.5820
Epoch 13/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3230081792.0000 - root_mean_squared_erro
r: 56827.1680 - val_loss: 3720891648.0000 - val_root_mean_squared_error: 60999.1133
Epoch 14/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 3200335104.0000 - root_mean_squared_erro
r: 56560.8555 - val_loss: 3624494592.0000 - val_root_mean_squared_error: 60203.7773
Epoch 15/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3000845568.0000 - root_mean_squared_erro
r: 54776.1641 - val_loss: 3475805440.0000 - val_root_mean_squared_error: 58955.9609
Epoch 16/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 3199978496.0000 - root_mean_squared_erro
r: 56555.0469 - val_loss: 3342490624.0000 - val_root_mean_squared_error: 57814.2773
Epoch 17/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 3133114624.0000 - root_mean_squared_erro
r: 55958.9961 - val_loss: 3940275200.0000 - val_root_mean_squared_error: 62771.6133
Epoch 18/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2926949632.0000 - root_mean_squared_erro
r: 54083.2812 - val_loss: 3205490176.0000 - val_root_mean_squared_error: 56617.0469
Epoch 19/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2936738560.0000 - root_mean_squared_erro
r: 54177.9844 - val_loss: 3238283776.0000 - val_root_mean_squared_error: 56905.9219
Epoch 20/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2983696896.0000 - root_mean_squared_erro
r: 54601.5195 - val_loss: 3240989696.0000 - val_root_mean_squared_error: 56929.6914
Epoch 21/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2800329216.0000 - root_mean_squared_erro
r: 52897.8438 - val_loss: 3930374144.0000 - val_root_mean_squared_error: 62692.6953
Epoch 22/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2830406656.0000 - root_mean_squared_erro
r: 53186.4414 - val_loss: 3484327936.0000 - val_root_mean_squared_error: 59028.1953
Epoch 23/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2929417472.0000 - root_mean_squared_erro
r: 54115.7500 - val_loss: 3241837568.0000 - val_root_mean_squared_error: 56937.1367
Epoch 24/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2801490432.0000 - root_mean_squared_erro
r: 52905.1016 - val_loss: 3053377536.0000 - val_root_mean_squared_error: 55257.3750
Epoch 25/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2815137280.0000 - root_mean_squared_erro
r: 53031.0039 - val_loss: 3200846080.0000 - val_root_mean_squared_error: 56576.0195
Epoch 26/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2819328000.0000 - root_mean_squared_erro
r: 53092.1250 - val_loss: 3417561088.0000 - val_root_mean_squared_error: 58459.9102
Epoch 27/100
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2726735360.0000 - root_mean_squared_erro
r: 52211.3008 - val_loss: 3274870016.0000 - val_root_mean_squared_error: 57226.4805
Epoch 28/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2852363264.0000 - root_mean_squared_erro
r: 53390.0195 - val_loss: 3414737920.0000 - val_root_mean_squared_error: 58435.7578
Epoch 29/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2770728192.0000 - root_mean_squared_erro
r: 52624.4492 - val_loss: 3434776576.0000 - val_root_mean_squared_error: 58606.9688
Epoch 30/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2789532160.0000 - root_mean_squared_erro
r: 52802.5547 - val_loss: 3262315520.0000 - val_root_mean_squared_error: 57116.6836
Epoch 31/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2683632896.0000 - root_mean_squared_erro
r: 51798.2422 - val_loss: 3737213440.0000 - val_root_mean_squared_error: 61132.7539
Epoch 32/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2675304192.0000 - root_mean_squared_erro
r: 51712.0391 - val_loss: 3171623168.0000 - val_root_mean_squared_error: 56317.1641
Epoch 33/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2906547200.0000 - root_mean_squared_erro
r: 53903.5508 - val_loss: 3870891520.0000 - val_root_mean_squared_error: 62216.4883
Epoch 34/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2692177920.0000 - root_mean_squared_erro
r: 51868.2695 - val_loss: 3158903296.0000 - val_root_mean_squared_error: 56204.1211
Epoch 35/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2738666496.0000 - root_mean_squared_erro
r: 52324.6719 - val_loss: 3229836288.0000 - val_root_mean_squared_error: 56831.6484
Epoch 36/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2649613568.0000 - root_mean_squared_erro
r: 51456.8047 - val_loss: 3121056768.0000 - val_root_mean_squared_error: 55866.4180
Epoch 37/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2725374464.0000 - root_mean_squared_erro
r: 52188.0586 - val_loss: 3146051840.0000 - val_root_mean_squared_error: 56089.6758
Epoch 38/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2729904640.0000 - root_mean_squared_erro
r: 52244.3594 - val_loss: 3123862528.0000 - val_root_mean_squared_error: 55891.5234
Epoch 39/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 2681320192.0000 - root_mean_squared_erro
r: 51773.8789 - val_loss: 3806076160.0000 - val_root_mean_squared_error: 61693.4062
Epoch 40/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2695783424.0000 - root_mean_squared_erro
r: 51904.8789 - val_loss: 3033415680.0000 - val_root_mean_squared_error: 55076.4531
Epoch 41/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2676339200.0000 - root_mean_squared_erro
r: 51722.9922 - val_loss: 4016765696.0000 - val_root_mean_squared_error: 63377.9609
Epoch 42/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2690556160.0000 - root_mean_squared_erro
r: 51855.9062 - val_loss: 3145768448.0000 - val_root_mean_squared_error: 56087.1523
Epoch 43/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2669172480.0000 - root_mean_squared_erro
r: 51655.9688 - val_loss: 3064726016.0000 - val_root_mean_squared_error: 55359.9688
Epoch 44/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2628993536.0000 - root_mean_squared_erro
r: 51267.3164 - val_loss: 3216302592.0000 - val_root_mean_squared_error: 56712.4570
Epoch 45/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2689366016.0000 - root_mean_squared_erro
r: 51845.2773 - val_loss: 3065277952.0000 - val_root_mean_squared_error: 55364.9531
Epoch 46/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2628811776.0000 - root_mean_squared_erro
r: 51264.6992 - val_loss: 3089552896.0000 - val_root_mean_squared_error: 55583.7461
Epoch 47/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2566642688.0000 - root_mean_squared_erro
r: 50640.9062 - val_loss: 3068019200.0000 - val_root_mean_squared_error: 55389.7031
Epoch 48/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2688141312.0000 - root_mean_squared_erro
r: 51812.4023 - val_loss: 3369011968.0000 - val_root_mean_squared_error: 58043.1914
Epoch 49/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2676328960.0000 - root_mean_squared_erro
r: 51730.8008 - val_loss: 2920869888.0000 - val_root_mean_squared_error: 54045.0742
Epoch 50/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2496260864.0000 - root_mean_squared_erro
r: 49948.8711 - val_loss: 3018246912.0000 - val_root_mean_squared_error: 54938.5742
Epoch 51/100
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2580494336.0000 - root_mean_squared_erro
r: 50791.3164 - val_loss: 2951595264.0000 - val_root_mean_squared_error: 54328.5859
Epoch 52/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2675419392.0000 - root_mean_squared_erro
r: 51707.8516 - val_loss: 3352327424.0000 - val_root_mean_squared_error: 57899.2852
Epoch 53/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2552685824.0000 - root_mean_squared_erro
r: 50503.9609 - val_loss: 3785790976.0000 - val_root_mean_squared_error: 61528.7812
Epoch 54/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 2557549824.0000 - root_mean_squared_erro
r: 50567.8008 - val_loss: 3115501568.0000 - val_root_mean_squared_error: 55816.6797
Epoch 55/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 2514075392.0000 - root_mean_squared_erro
r: 50124.4102 - val_loss: 2995494144.0000 - val_root_mean_squared_error: 54731.1094
Epoch 56/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2553985536.0000 - root_mean_squared_erro
r: 50529.9766 - val_loss: 3158546688.0000 - val_root_mean_squared_error: 56200.9492
Epoch 57/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2590243584.0000 - root_mean_squared_erro
r: 50872.2852 - val_loss: 3221155072.0000 - val_root_mean_squared_error: 56755.2188
Epoch 58/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2505511680.0000 - root_mean_squared_erro
r: 50037.0547 - val_loss: 3064176384.0000 - val_root_mean_squared_error: 55355.0039
Epoch 59/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2668993792.0000 - root_mean_squared_erro
r: 51649.0625 - val_loss: 3111876096.0000 - val_root_mean_squared_error: 55784.1914
Epoch 60/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2455049728.0000 - root_mean_squared_erro
r: 49526.9258 - val_loss: 3074504704.0000 - val_root_mean_squared_error: 55448.2148
Epoch 61/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2616104192.0000 - root_mean_squared_erro
r: 51141.8477 - val_loss: 3059302912.0000 - val_root_mean_squared_error: 55310.9648
Epoch 62/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 2450646528.0000 - root_mean_squared_erro
r: 49488.2734 - val_loss: 3288945152.0000 - val_root_mean_squared_error: 57349.3242
Epoch 63/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2582804992.0000 - root_mean_squared_erro
r: 50816.5664 - val_loss: 3403234560.0000 - val_root_mean_squared_error: 58337.2500
Epoch 64/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2603771904.0000 - root_mean_squared_erro
r: 51014.6523 - val_loss: 3203316736.0000 - val_root_mean_squared_error: 56597.8516
Epoch 65/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 2639634944.0000 - root_mean_squared_erro
r: 51366.7812 - val_loss: 3038631168.0000 - val_root_mean_squared_error: 55123.7812
Epoch 66/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2516060160.0000 - root_mean_squared_erro
r: 50153.4609 - val_loss: 2941582336.0000 - val_root_mean_squared_error: 54236.3555
Epoch 67/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2492644864.0000 - root_mean_squared_erro
r: 49915.3711 - val_loss: 3040054016.0000 - val_root_mean_squared_error: 55136.6836
Epoch 68/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 2482366208.0000 - root_mean_squared_erro
r: 49817.3281 - val_loss: 3214752512.0000 - val_root_mean_squared_error: 56698.7891
Epoch 69/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2425843968.0000 - root_mean_squared_erro
r: 49240.5664 - val_loss: 3042396672.0000 - val_root_mean_squared_error: 55157.9258
Epoch 70/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2462455552.0000 - root_mean_squared_erro
r: 49595.4336 - val_loss: 3221079552.0000 - val_root_mean_squared_error: 56754.5547
Epoch 71/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2454253312.0000 - root_mean_squared_erro
r: 49530.9531 - val_loss: 3042770944.0000 - val_root_mean_squared_error: 55161.3164
Epoch 72/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2423707904.0000 - root_mean_squared_erro
r: 49211.4297 - val_loss: 3126023168.0000 - val_root_mean_squared_error: 55910.8516
Epoch 73/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2559433984.0000 - root_mean_squared_erro
r: 50567.7109 - val_loss: 3018620160.0000 - val_root_mean_squared_error: 54941.9727
Epoch 74/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 2611820544.0000 - root_mean_squared_erro
r: 51095.9570 - val_loss: 3227091200.0000 - val_root_mean_squared_error: 56807.4922
Epoch 75/100
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 2554323456.0000 - root_mean_squared_erro
r: 50532.6523 - val_loss: 2990388480.0000 - val_root_mean_squared_error: 54684.4453
Epoch 76/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2524417024.0000 - root_mean_squared_erro
r: 50216.8281 - val_loss: 3016230144.0000 - val_root_mean_squared_error: 54920.2148
Epoch 77/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2562827264.0000 - root_mean_squared_erro
r: 50601.2383 - val_loss: 3110782208.0000 - val_root_mean_squared_error: 55774.3867
Epoch 78/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2510818048.0000 - root_mean_squared_erro
r: 50095.9805 - val_loss: 3107752704.0000 - val_root_mean_squared_error: 55747.2227
Epoch 79/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2418304512.0000 - root_mean_squared_erro
r: 49171.0078 - val_loss: 2953392896.0000 - val_root_mean_squared_error: 54345.1289
Epoch 80/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2422451456.0000 - root_mean_squared_erro
r: 49216.2734 - val_loss: 3055053312.0000 - val_root_mean_squared_error: 55272.5352
Epoch 81/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 2455007488.0000 - root_mean_squared_erro
r: 49531.5781 - val_loss: 3045969408.0000 - val_root_mean_squared_error: 55190.3008
Epoch 82/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2419768576.0000 - root_mean_squared_erro
r: 49178.2539 - val_loss: 3074956544.0000 - val_root_mean_squared_error: 55452.2891
Epoch 83/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2475449856.0000 - root_mean_squared_erro
r: 49741.2188 - val_loss: 3053578240.0000 - val_root_mean_squared_error: 55259.1914
Epoch 84/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2460014848.0000 - root_mean_squared_erro
r: 49586.3242 - val_loss: 2930732032.0000 - val_root_mean_squared_error: 54136.2344
Epoch 85/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2466914816.0000 - root_mean_squared_erro
r: 49663.2266 - val_loss: 3089690624.0000 - val_root_mean_squared_error: 55584.9844
Epoch 86/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2354352896.0000 - root_mean_squared_erro
r: 48514.3359 - val_loss: 2972948480.0000 - val_root_mean_squared_error: 54524.7500
Epoch 87/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2393474560.0000 - root_mean_squared_erro
r: 48903.1094 - val_loss: 3066971392.0000 - val_root_mean_squared_error: 55380.2422
Epoch 88/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 2507817728.0000 - root_mean_squared_erro
r: 50065.5938 - val_loss: 3044221184.0000 - val_root_mean_squared_error: 55174.4609
Epoch 89/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2438450944.0000 - root_mean_squared_erro
r: 49375.7148 - val_loss: 2939404288.0000 - val_root_mean_squared_error: 54216.2734
Epoch 90/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2373002752.0000 - root_mean_squared_erro
r: 48706.0273 - val_loss: 3084823296.0000 - val_root_mean_squared_error: 55541.1875
Epoch 91/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 2367094272.0000 - root_mean_squared_erro
r: 48645.6484 - val_loss: 3107561216.0000 - val_root_mean_squared_error: 55745.5039
Epoch 92/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2442265344.0000 - root_mean_squared_erro
r: 49413.9883 - val_loss: 3187333632.0000 - val_root_mean_squared_error: 56456.4766
Epoch 93/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2420281088.0000 - root_mean_squared_erro
r: 49183.8438 - val_loss: 3003081728.0000 - val_root_mean_squared_error: 54800.3789
Epoch 94/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 2492461312.0000 - root_mean_squared_erro
r: 49917.5781 - val_loss: 2895421952.0000 - val_root_mean_squared_error: 53809.1250
Epoch 95/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 2491831296.0000 - root_mean_squared_erro
r: 49912.1211 - val_loss: 3070292992.0000 - val_root_mean_squared_error: 55410.2266
Epoch 96/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2385057024.0000 - root_mean_squared_erro
r: 48829.8594 - val_loss: 3248151296.0000 - val_root_mean_squared_error: 56992.5547
Epoch 97/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2567180032.0000 - root_mean_squared_erro
r: 50659.0664 - val_loss: 3005500416.0000 - val_root_mean_squared_error: 54822.4453
Epoch 98/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2410853632.0000 - root_mean_squared_erro
r: 49095.7734 - val_loss: 3089205248.0000 - val_root_mean_squared_error: 55580.6211
Epoch 99/100
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2425818880.0000 - root_mean_squared_erro
r: 49246.5039 - val_loss: 3288166400.0000 - val_root_mean_squared_error: 57342.5352
Epoch 100/100
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 2372902144.0000 - root_mean_squared_erro
r: 48691.7617 - val_loss: 2922787584.0000 - val_root_mean_squared_error: 54062.8125
```

Out[ ]:

```
<keras.src.callbacks.history.History at 0x788062620e80>
```

In [ ]:

```
large_nn = load_model('models/large_nn.keras')
mse(large_nn.predict(X_train), y_train, squared=False), mse(large_nn.predict(X_val), y_v
al, squared=False)
```

```
563/563 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
39/39 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
(47861.855, 53809.13)
```

In [ ]:

```
mse(gbr.predict(X_test), y_test, squared=False)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_regression.py:492: FutureWarning
: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the roo
t mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

Out[ ]:

```
49345.935411338134
```

Evaluation

Models are evaluated using the Root Mean Squared Error (RMSE):

1)Linear Regression: Poor performance due to limited complexity.

2)KNN: Improved accuracy but risks overfitting with fewer neighbors.

3)Random Forest: Strong performance with manageable complexity.

4)Gradient Boosting: Best among traditional models, achieving low RMSE.

5)Neural Networks: Increasing complexity improves accuracy, but the risk of overfitting rises with deeper architectures.

CONCLUSION

The most suitable Model here is Gradient Boosting achieved the best overall performance on the test set with the lowest RMSE (~49,000).

while Simpler models (like Linear Regression) are less effective for complex datasets.

Ensemble methods like Random Forest and Gradient Boosting excel in capturing non-linear relationships. and

Neural networks require careful tuning to balance complexity and overfitting. This project highlights the importance of trying multiple models, preprocessing data effectively, and evaluating results to make informed decisions.

decisions.