

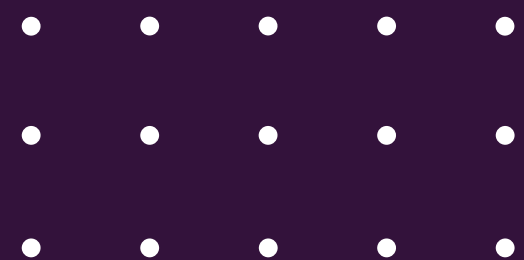


MOBILE PRICE PREDICTION

GROUP-05

OUTLINE

- Brief about Mobile Price Prediction
- Requirement of Mobile Price Prediction
- About the dataset
- Machine Learning Process
- Selection of Model
- Model Prediction

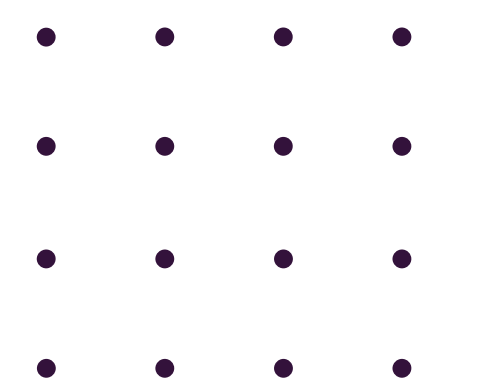
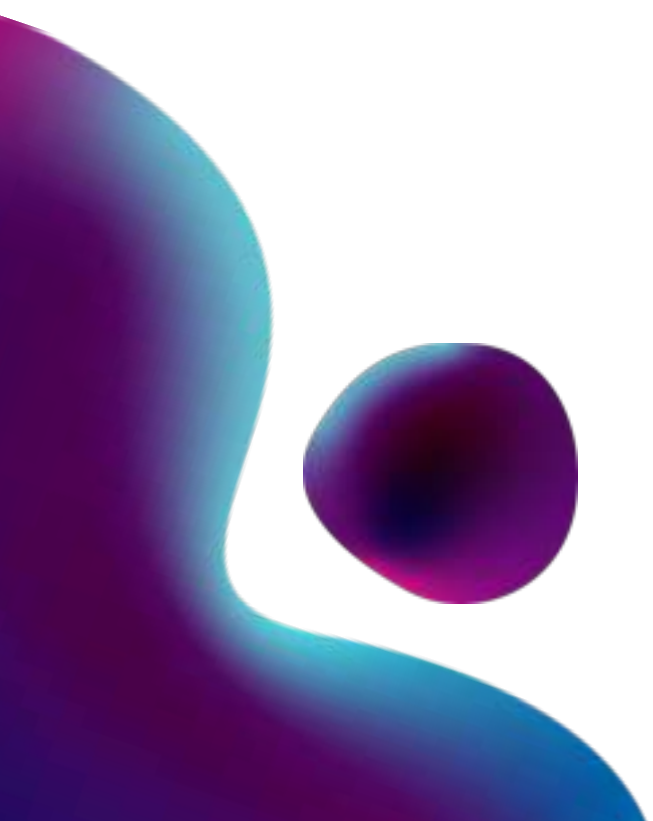




MOBILE PRICE PREDICTION

Phone price prediction refers to the process of using machine learning or statistical models to predict the price of a mobile phone based on various features or specifications.

Project typically involves:

- Data Collection
 - Data Processing
 - Features Selection
 - Model Training
 - Testing and validation
 - Prediction
- 
- 

REQUIREMENT OF MOBILE PRICE PREDICTION?



- **Market Analysis:** Helps businesses and consumers understand price trends and make informed decisions.
- **Pricing Strategy:** Assists manufacturers in setting competitive prices for new models.
- **Consumer Assistance:** Aids buyers in evaluating if a phone is priced fairly based on its specs.
- **Resale Value Estimation:** Predicts the resale value of used phones.
- **E-commerce Recommendations:** Enhances online platforms by providing price estimates for better customer experience.



LET'S ANALYSE THE DATASET

ABOUT THE DATASET

	Ratings	RAM	ROM	Mobile_Size	Primary_Cam	Selfi_Cam	Battery_Power	Price
0	4.3	4.0	128.0	6.00	48	13.0	4000	24999
1	3.4	6.0	64.0	4.50	48	12.0	4000	15999
2	4.3	4.0	4.0	4.50	64	16.0	4000	15000
3	4.4	6.0	64.0	6.40	48	15.0	3800	18999
4	4.5	6.0	128.0	6.18	35	15.0	3800	18999

The dataset for this phone price prediction project includes attributes like RAM, storage, processor, camera quality, screen size, and brand, along with their market prices. This data is used to build a model that predicts phone prices based on these features.

MACHINE LEARNING PROCESS

Now comes the process modeling after we thoroughly analysed the dataset. Here are the key steps in machine learning modeling:

- Data Collection
- Importing of libraries & creating Dataframe
- Data Preprocessing
- Handling Outliers
- Data Splitting
- Feature Scaling

DATA COLLECTION

Data collection involves gathering detailed information on mobile phones, including features like RAM, storage, processor, and brand, along with their market prices. This data is essential for training the predictive model.

IMPORTING LIBRARIES



```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

NumPy 



Pandas

Library used for data manipulation, analysis, and handling structured data.

LIBRARIES



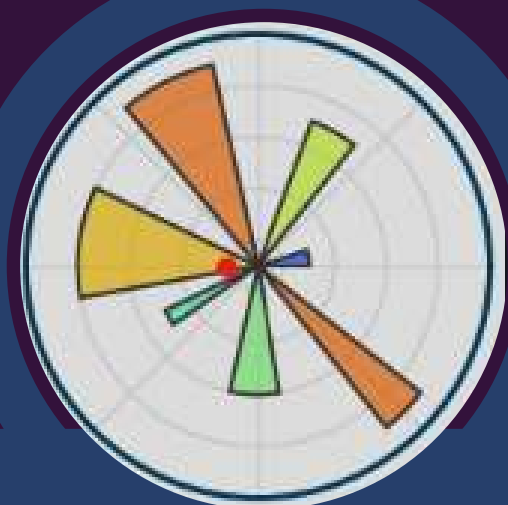
Numpy

Library for numerical computing, enabling efficient operations on arrays and matrices.



Seaborn

Library for creating visually appealing statistical graphics and data visualizations.



Matplotlib

Library for creating static, interactive, and animated visualizations and plots.

COLUMNS & DATATYPE

```
Index(['Ratings', 'RAM', 'ROM',  
      'Mobile_Size', 'Primary_Cam', 'Selfi_Cam',  
      'Battery_Power', 'Price'],  
      dtype='object')
```

```
df.head()  
df.dtypes
```

Ratings	float64
RAM	float64
ROM	float64
Mobile_Size	float64
Primary_Cam	int64
Selfi_Cam	float64
Battery_Power	int64
Price	int64
dtype: object	

df['Mobile_Size'] = df['Mobile_Size'].apply(lambda x: 6.5 if x>7 else x):

This code changes any `Mobile_Size` over 7 to 6.5, leaving smaller sizes the same.

df.describe(): This code gives a summary of numerical data, including count, mean, and range of values.

df['Mobile_Size'].unique() : Lists all the unique mobile sizes present in the dataset after the modification.

```
df['Mobile_Size'] = df['Mobile_Size'].apply(lambda x: 6.5 if x>7 else x)
df.describe()
df['Mobile_Size'].unique()
```

```
array([6.    , 4.5   , 6.4   , 6.18  , 5.8   , 6.7   , 6.53  , 4.54  , 5.5   ,
        4.7   , 6.67  , 3.7   , 6.2   , 2.    , 4.77  , 5.4   , 4.52  , 6.5   ,
        6.39  , 4.4   , 5.99  , 6.1   , 5.    , 5.7   , 6.41  , 6.3   , 4.58  ,
        4.57  , 6.28  , 4.8   , 6.08  , 6.22  , 4.503, 6.26  , 6.38  , 5.84  ,
        6.35  , 6.6   , 7.    , 6.55  , 6.59  , 6.44  , 5.6   , 6.52  , 5.9   ,
        5.65  ])
```




Correlation

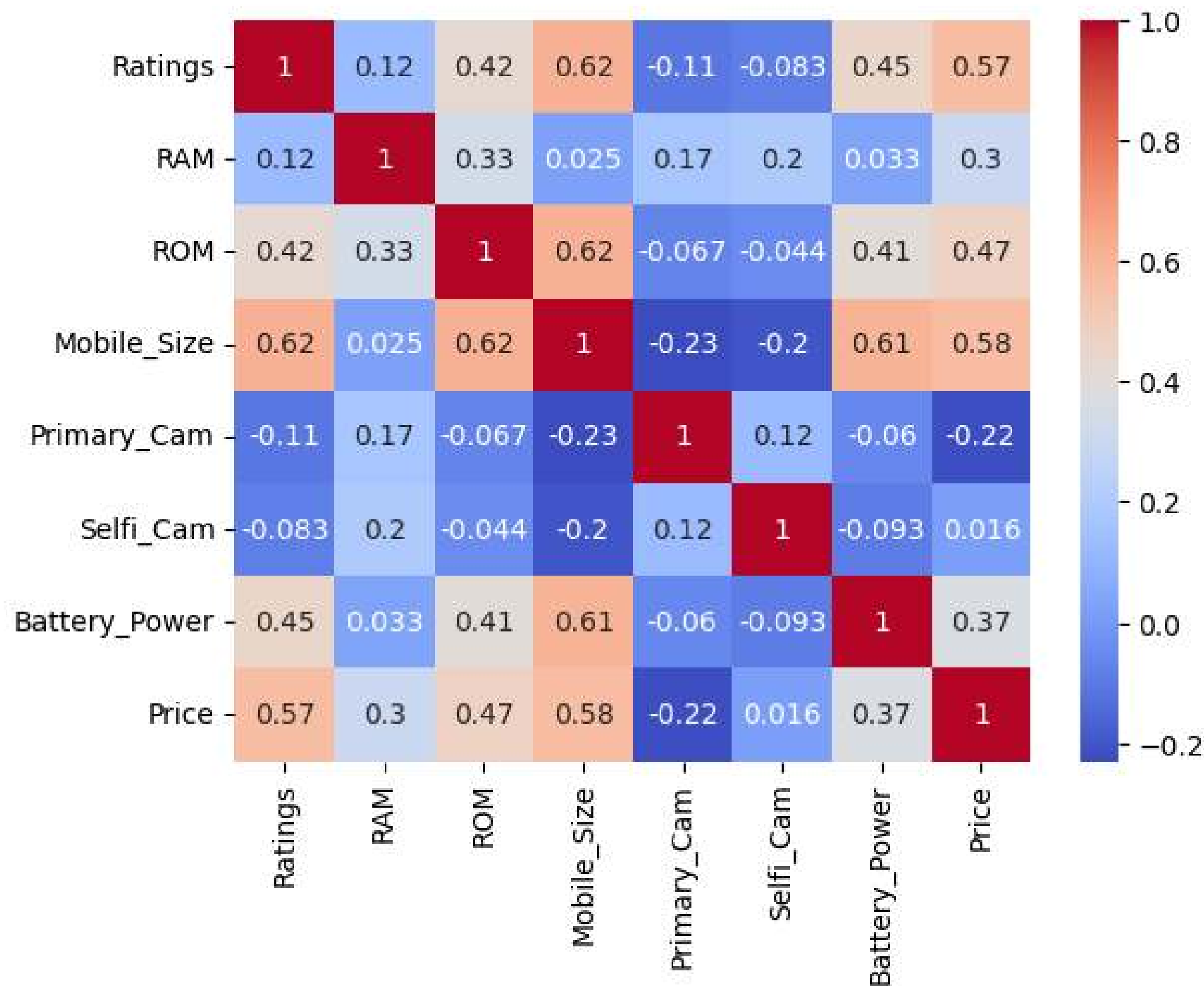
[, kor-ə-'lā-shən]

A statistic that measures the degree to which two securities move in relation to each other.

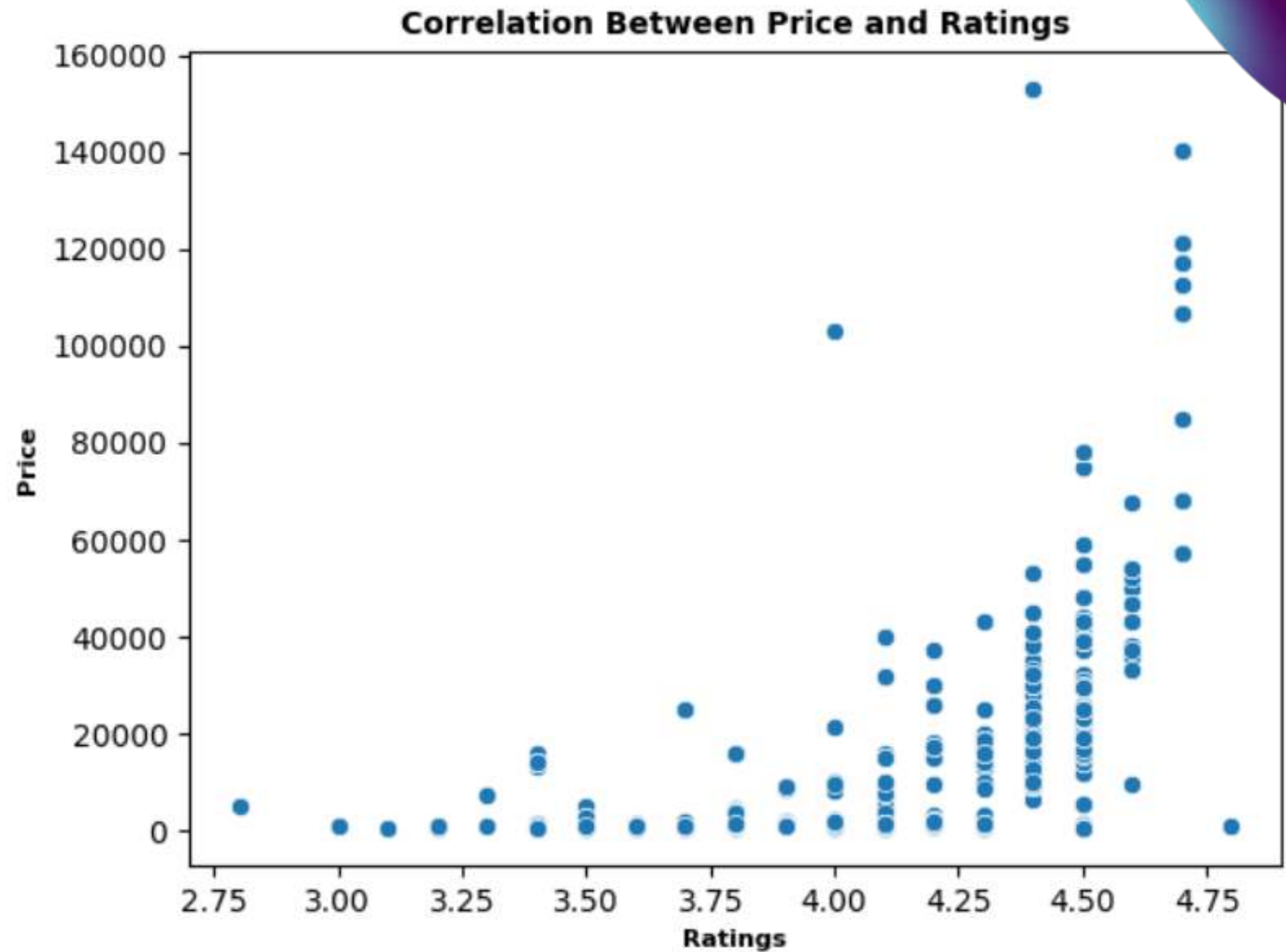
annot=True :

Adds the correlation values directly on the heatmap so you can see the exact numbers.

```
sns.heatmap(df.corr(), annot = True, cmap = 'coolwarm')  
plt.show()
```



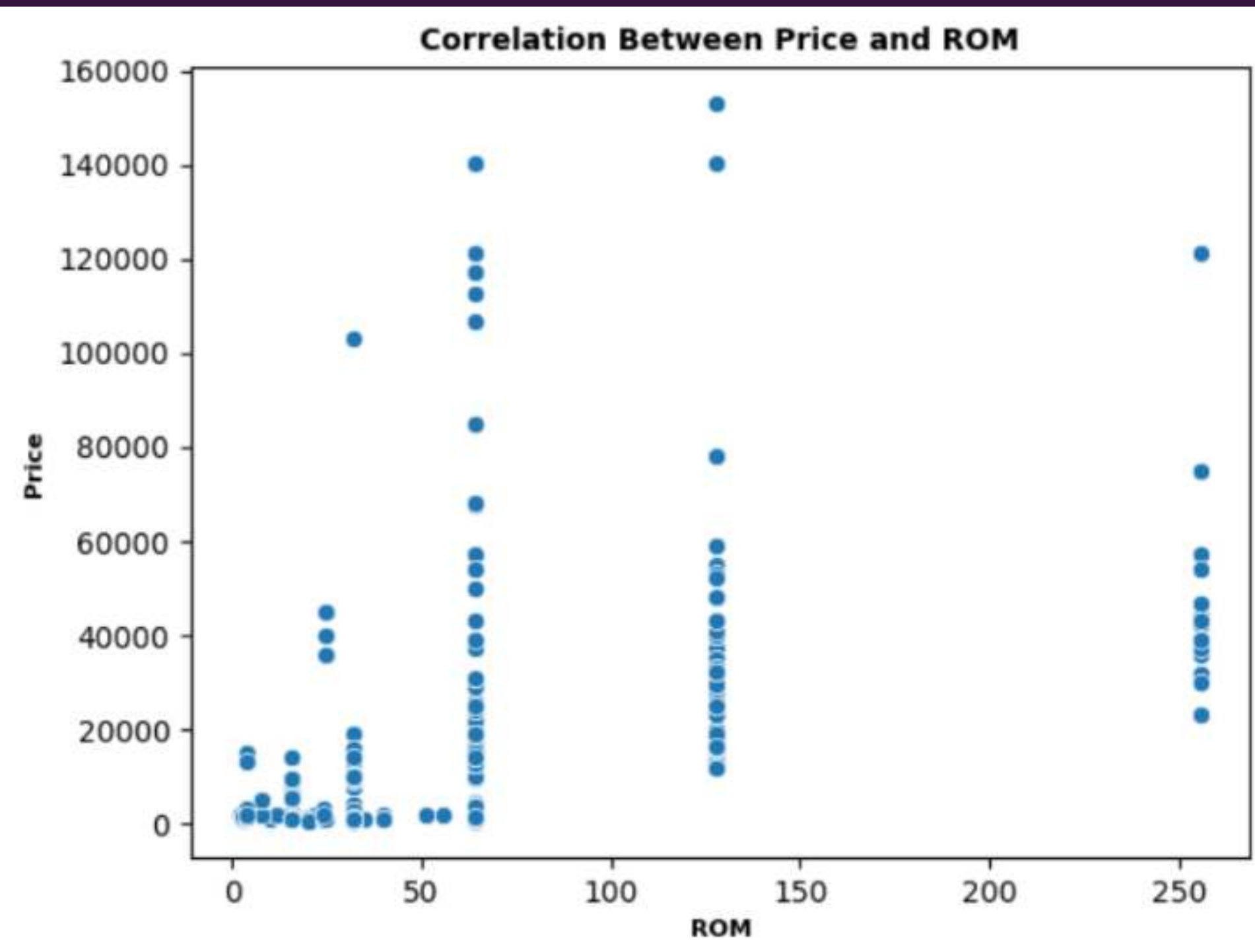
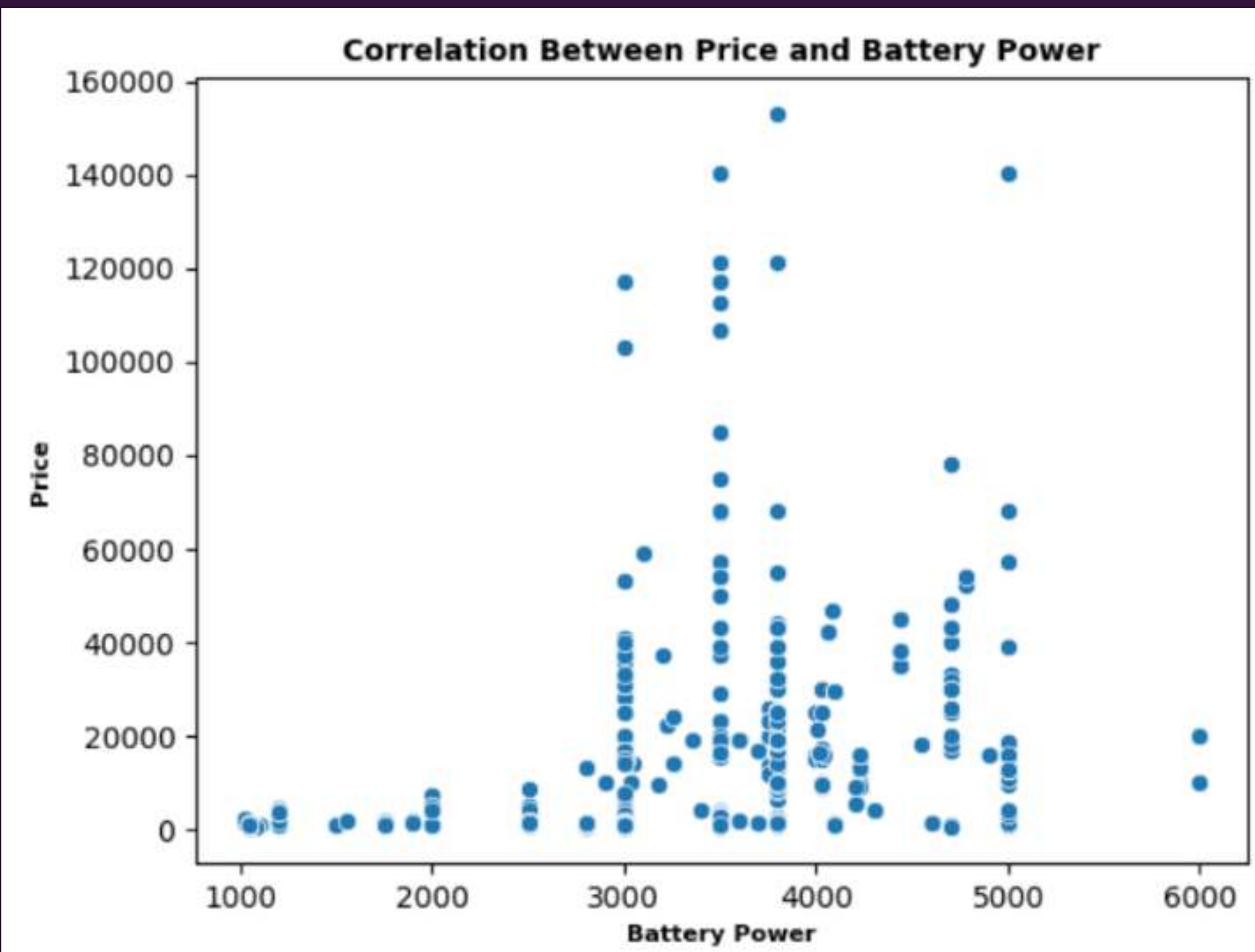
```
sns.scatterplot(x='Ratings',y='Price', data=df)
plt.xlabel('Ratings',weight='bold',size=8)
plt.ylabel('Price',weight='bold',size=8)
plt.title('Correlation Between Price and Ratings',weight='bold',size=10)
plt.show()
```



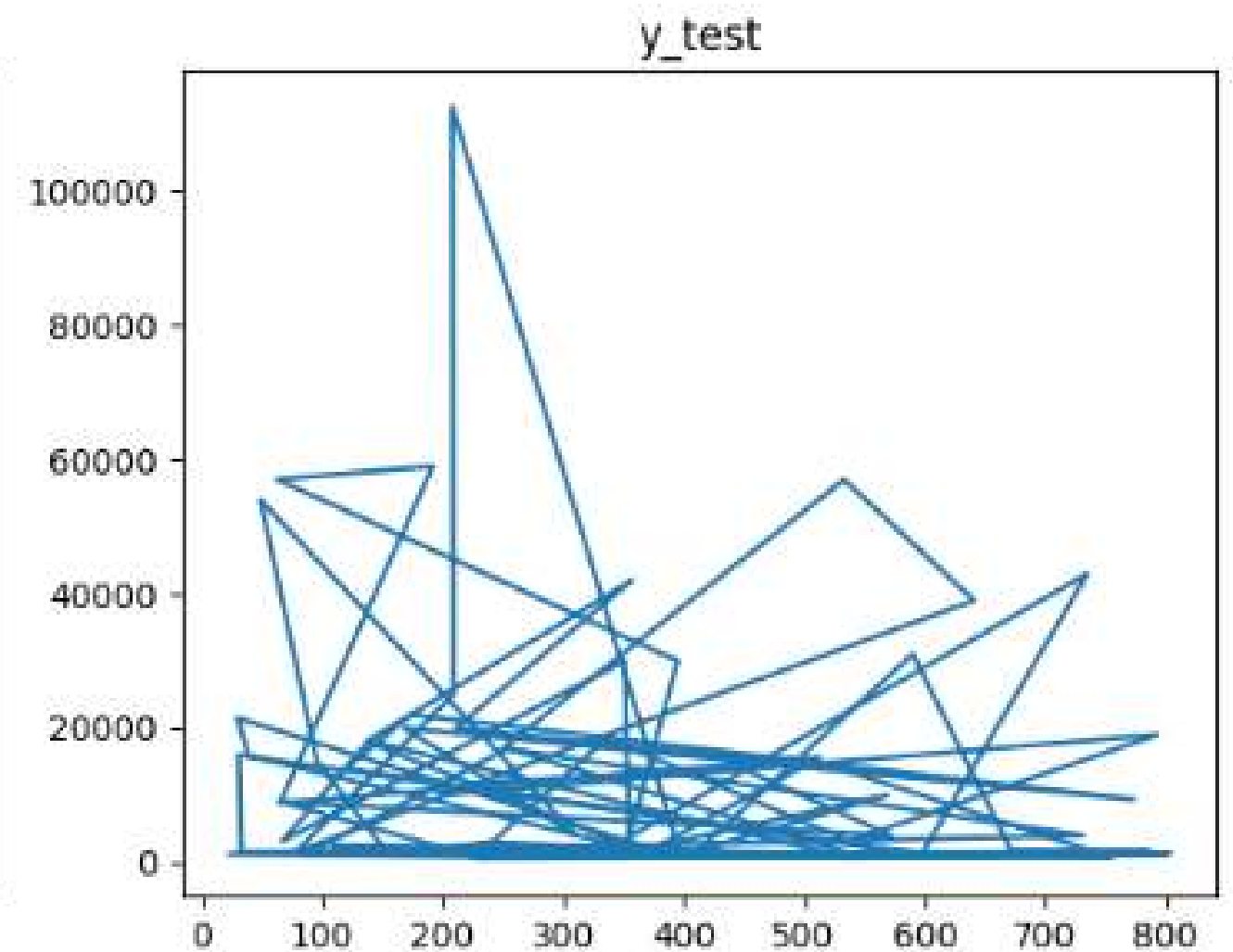
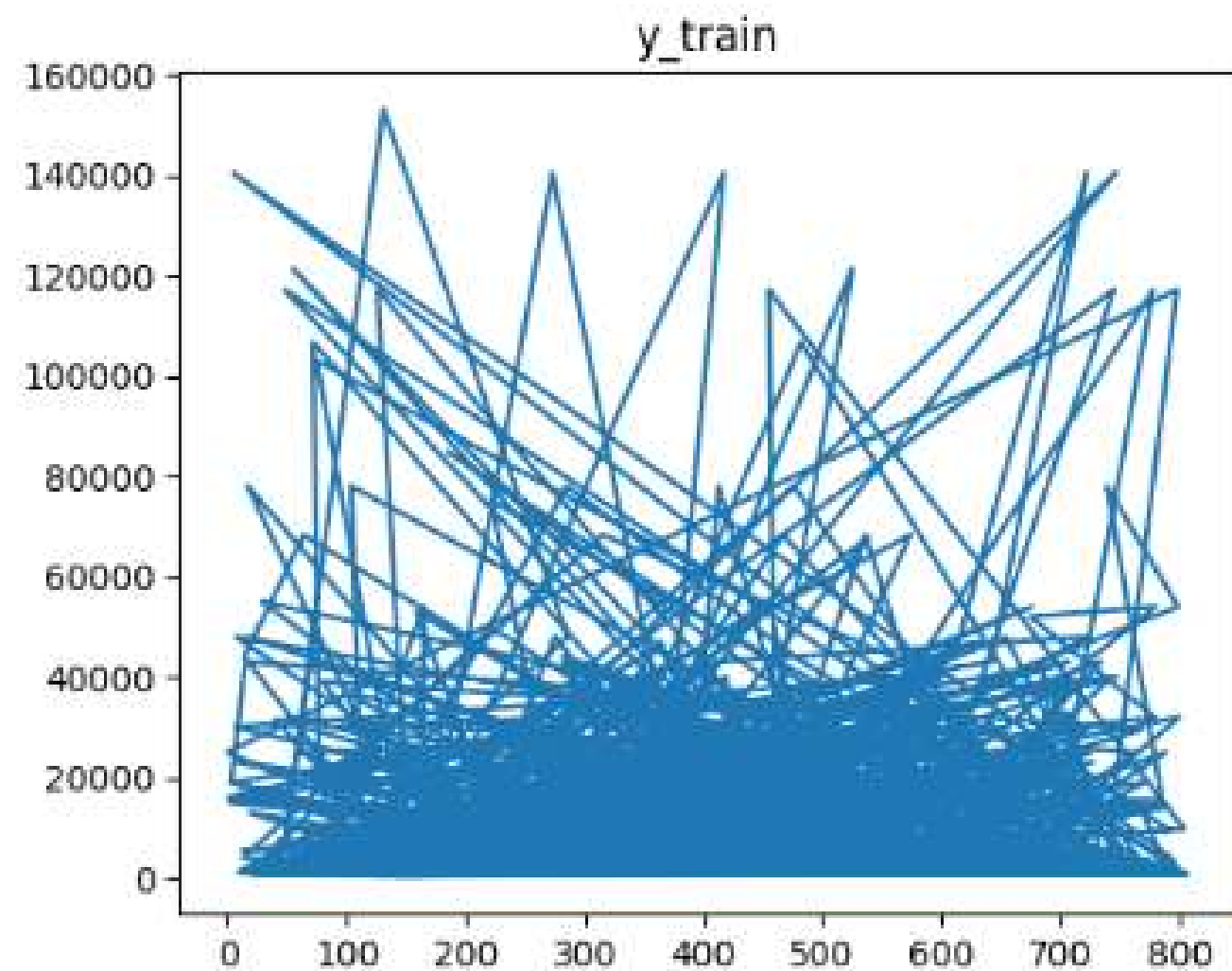
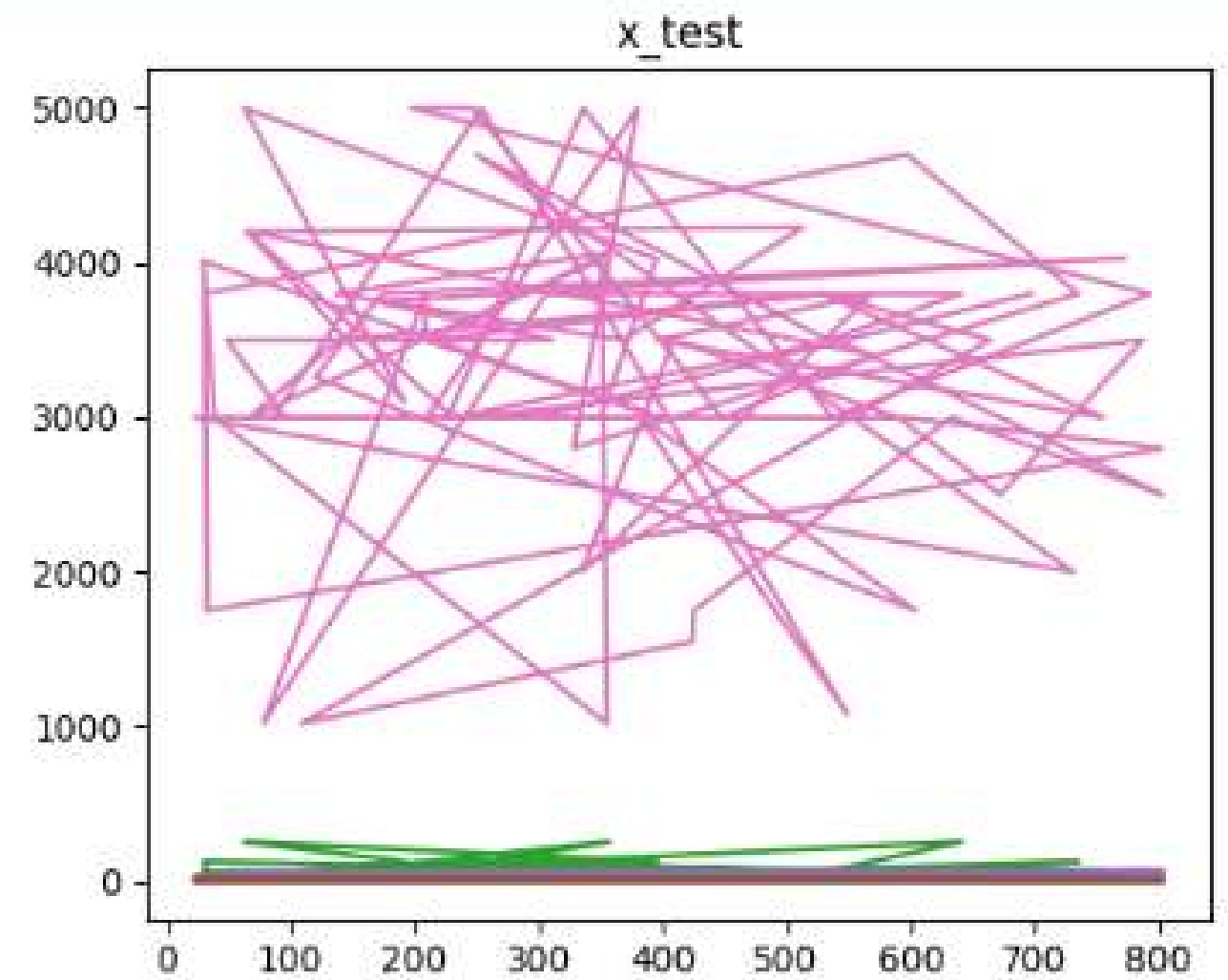
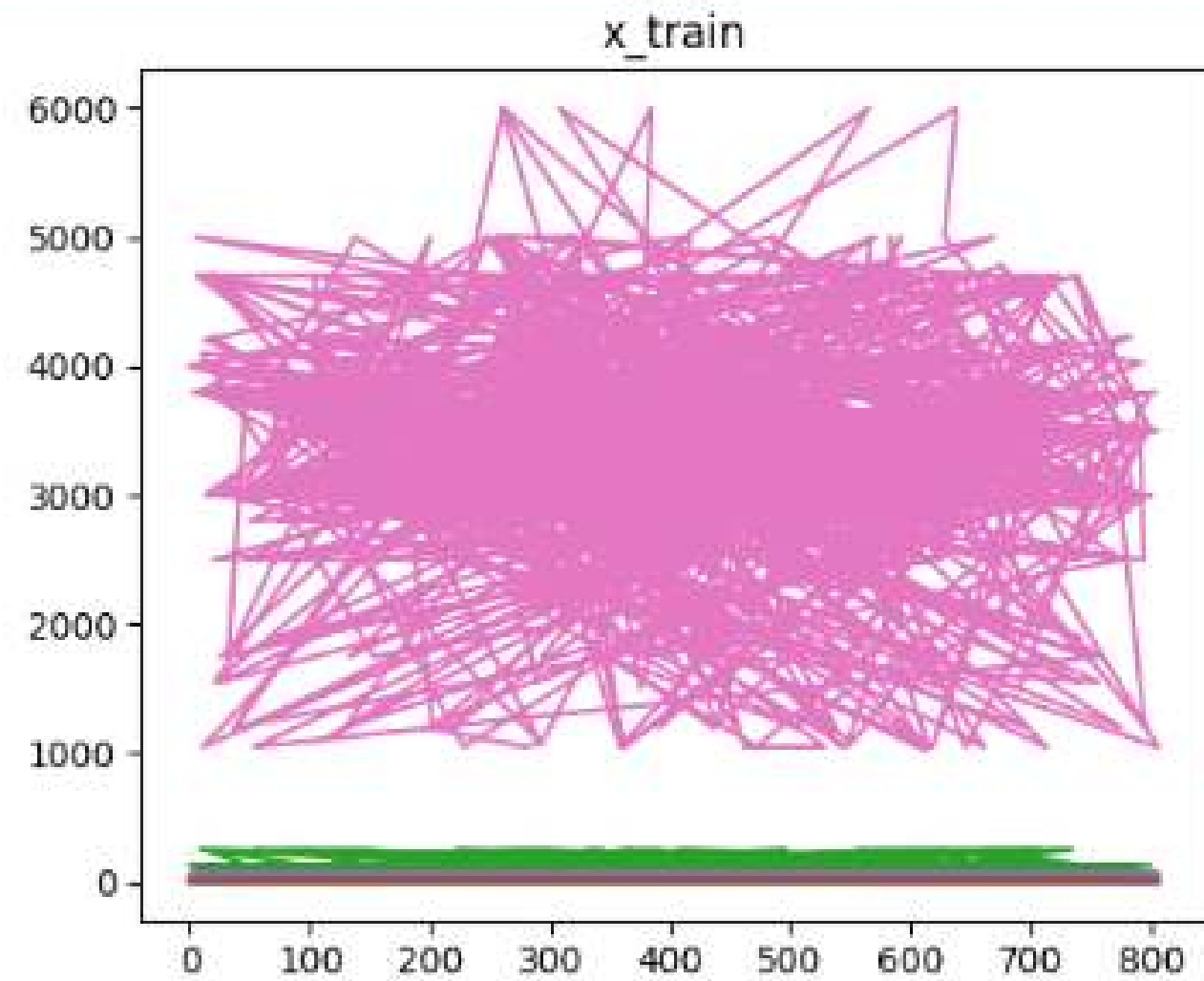
PRICE VS BATTERY POWER



PRICE VS ROM

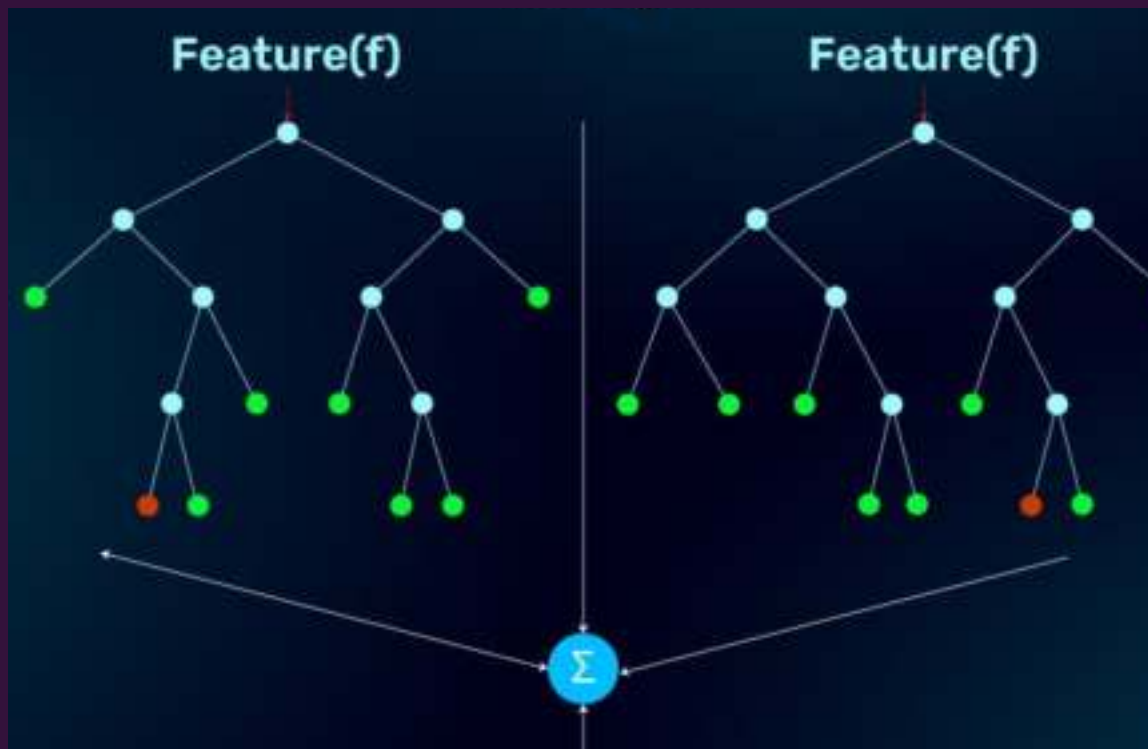


TRAINING & TESTING

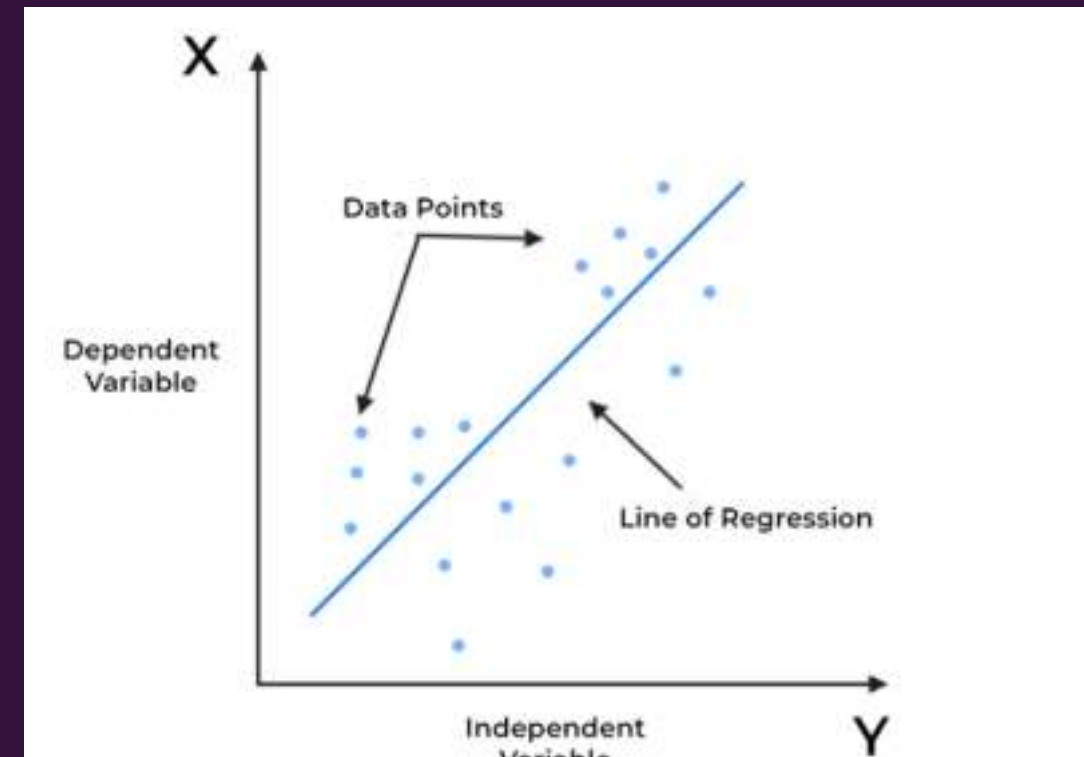


MODELS

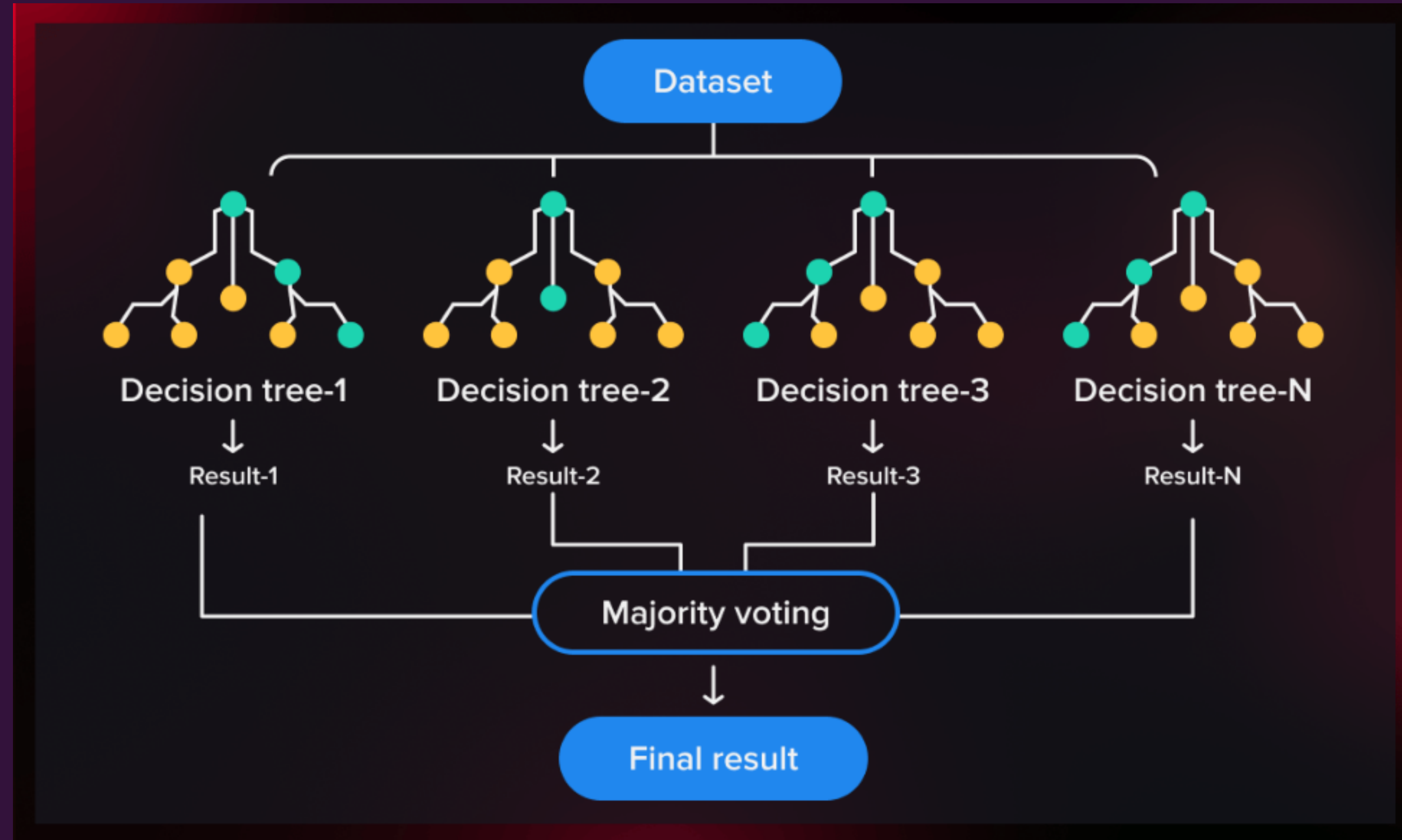
RANDOM FOREST REGRESSION



LINEAR REGRESSION



RANDOM FOREST



BRAND



Processor



Heat & Speed



CAMERA



**Design & Built
Quality**



**Power &
Storage**



Ask all friends then decide average by their predictions for a more accurate price estimate.



Decision Tree

It combines the predictions of many "trees" (my friends) to
come up with a final prediction



RANDOM FOREST REGRESSION

Random Forest Regression in machine learning is an ensemble (Ensemble means a group of elements viewed as a whole rather than individually) technique capable of performing both regression and classification tasks.

- **Versatile Prediction:** Random Forest Regression predicts numerical values by leveraging the strength of multiple decision trees.
- **Improved Accuracy:** It reduces overfitting and enhances accuracy by averaging the predictions of various trees in the forest.
- **Feature Importance:** Random Forest Regression can identify which features are most influential in predicting the target value, providing insights into the data.
- **Robustness:** It handles large datasets and high-dimensional data effectively, making it robust against noise and outliers.

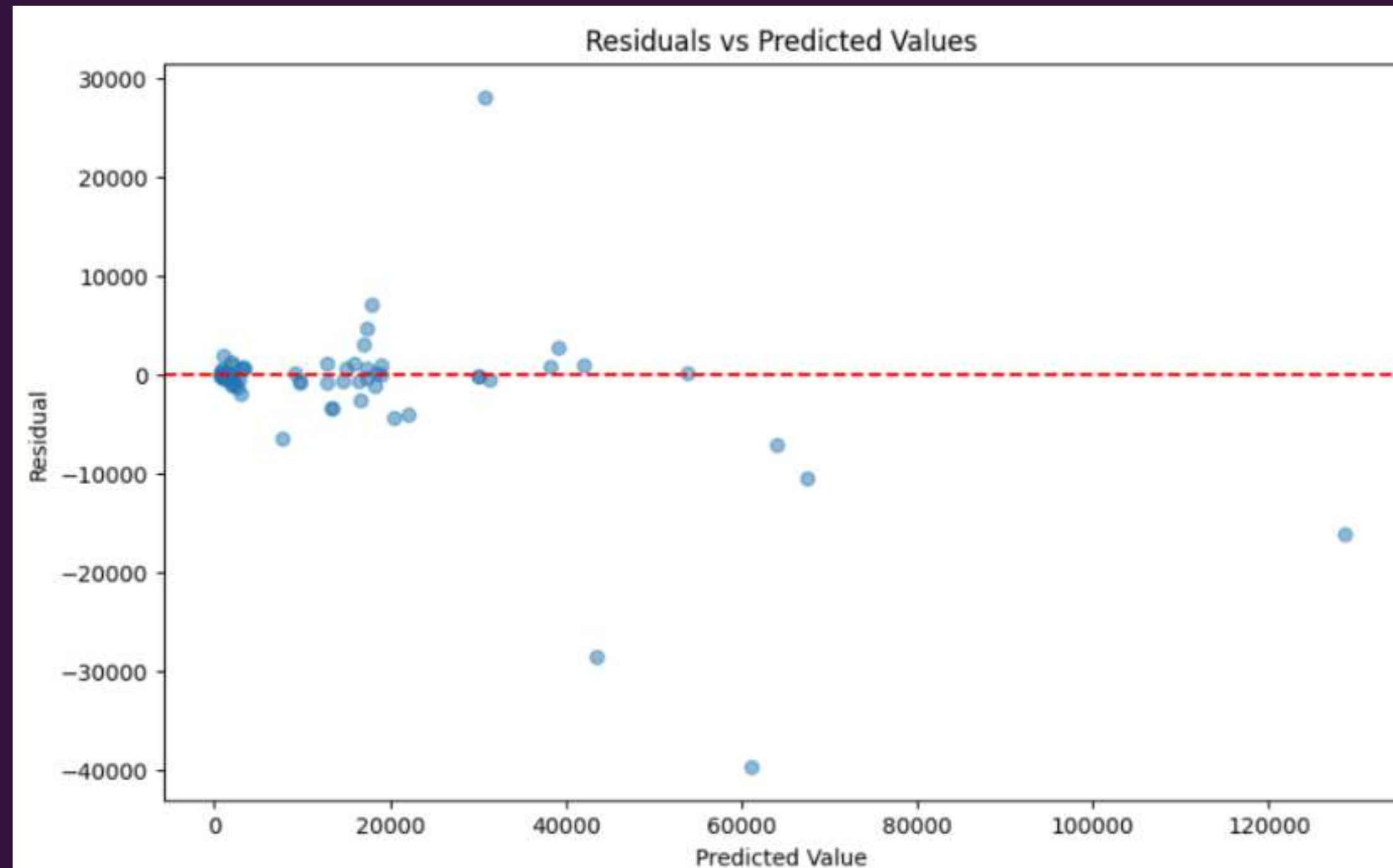
```
reg_model_diff = pd.DataFrame({'Actual value': y_test, 'Predicted value': y_pred})
reg_model_diff
```

Actual value		Actual value Predicted value	
697	1280	1280	1330.690000
296	985	985	963.394944
227	639	639	650.523810
336	18499	18499	18365.910000
537	3149	3149	2173.073333
...
597	1049	1049	2959.590000
735	42990	42990	42029.270000
328	982	982	1146.315000
380	1275	1275	7739.240000
210	1499	1499	1429.797000

RESIDUAL VALUE AND PREDICTED VALUE

Residual=Actual – Predicted

Predicted=Actual–Residual

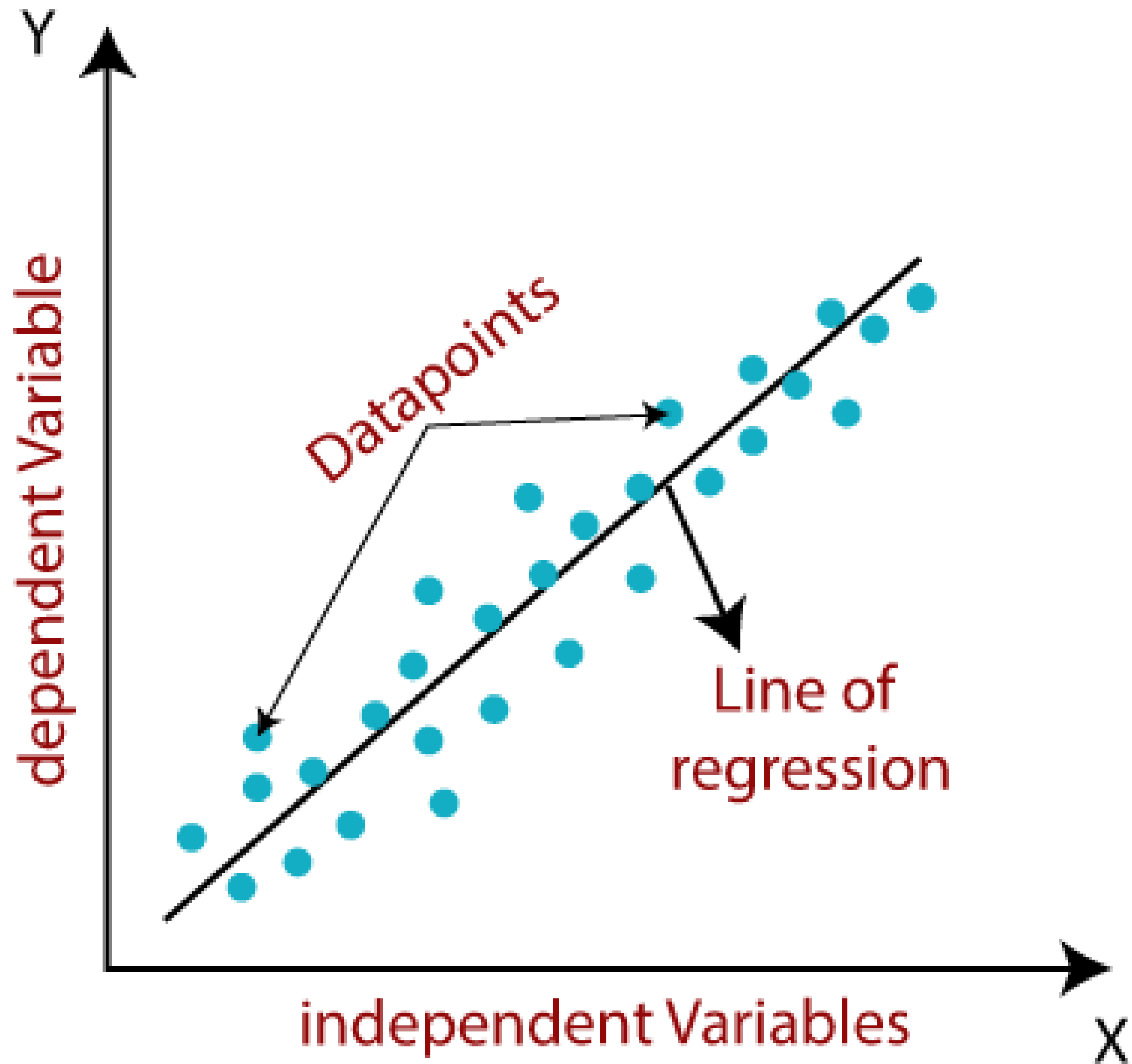




FUTURE SCOPE OF RANDOM FOREST

- **Enhanced Integration with Deep Learning:** Combining Random Forest with deep learning techniques to improve performance on complex problems.
- **Better Handling of Big Data:** Advancements could make Random Forest more efficient and scalable for even larger and more diverse datasets.
- **Increased Automation:** Future developments might automate the tuning of parameters and model selection, making it easier to use.
- **Improved Interpretability Tools:** New methods could emerge to help users better understand and interpret the results from Random Forest models.

LINEAR REGRESSION



Linear Regression Example



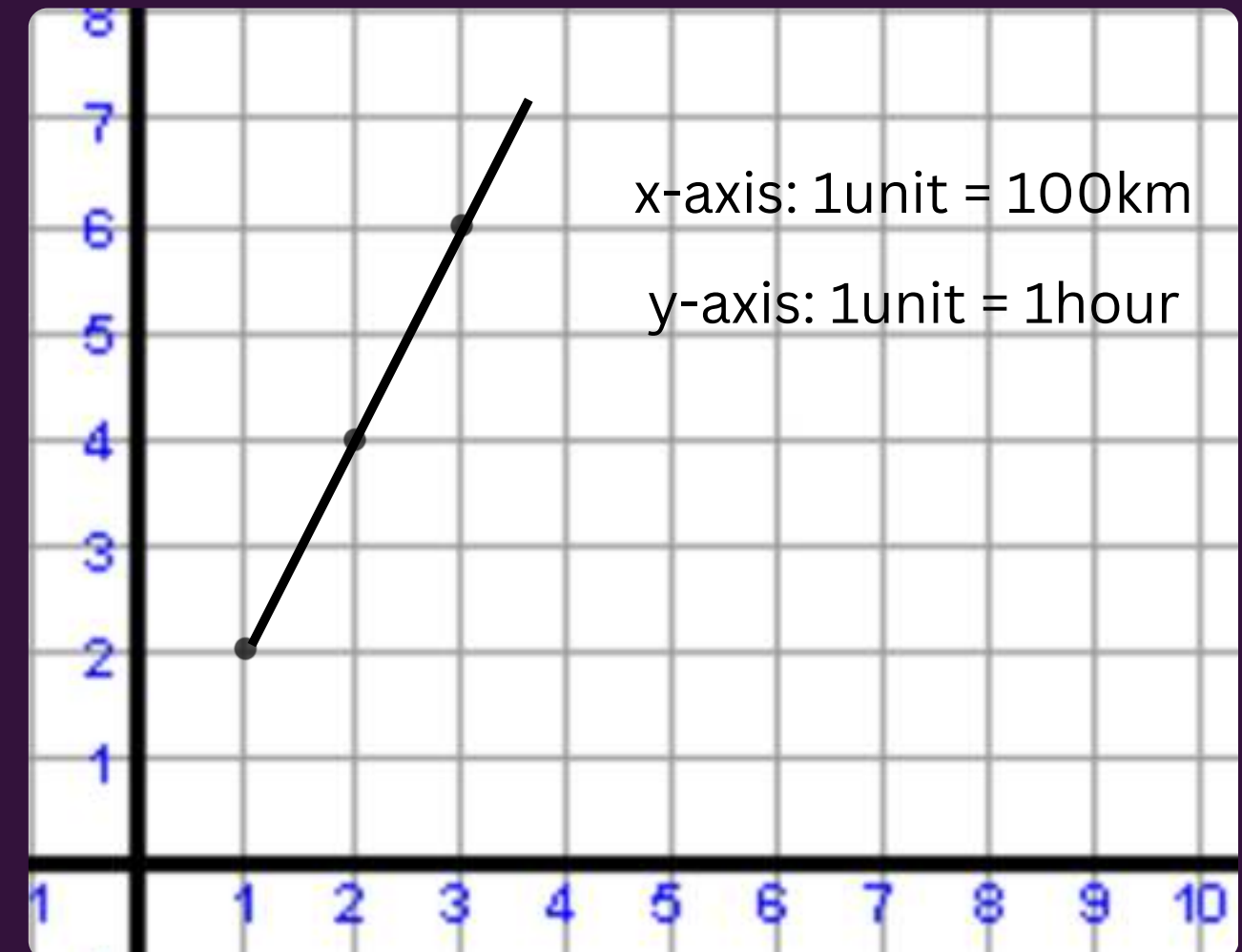
We notice that for every additional 100 km we drive, it takes about 2 more hours. Linear regression is like drawing a straight line through these points, illustrating that travel time increases proportionally with distance.

Past Trips:

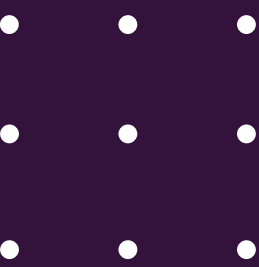
100 km: took 2 hours

200 km: took 4 hours

300 km: took 6 hours



Linear Regression



Linear regression is a method for predicting one variable based on another by finding the best-fitting straight line through a set of data points. This line represents the relationship between the variables and helps in making predictions about future values.

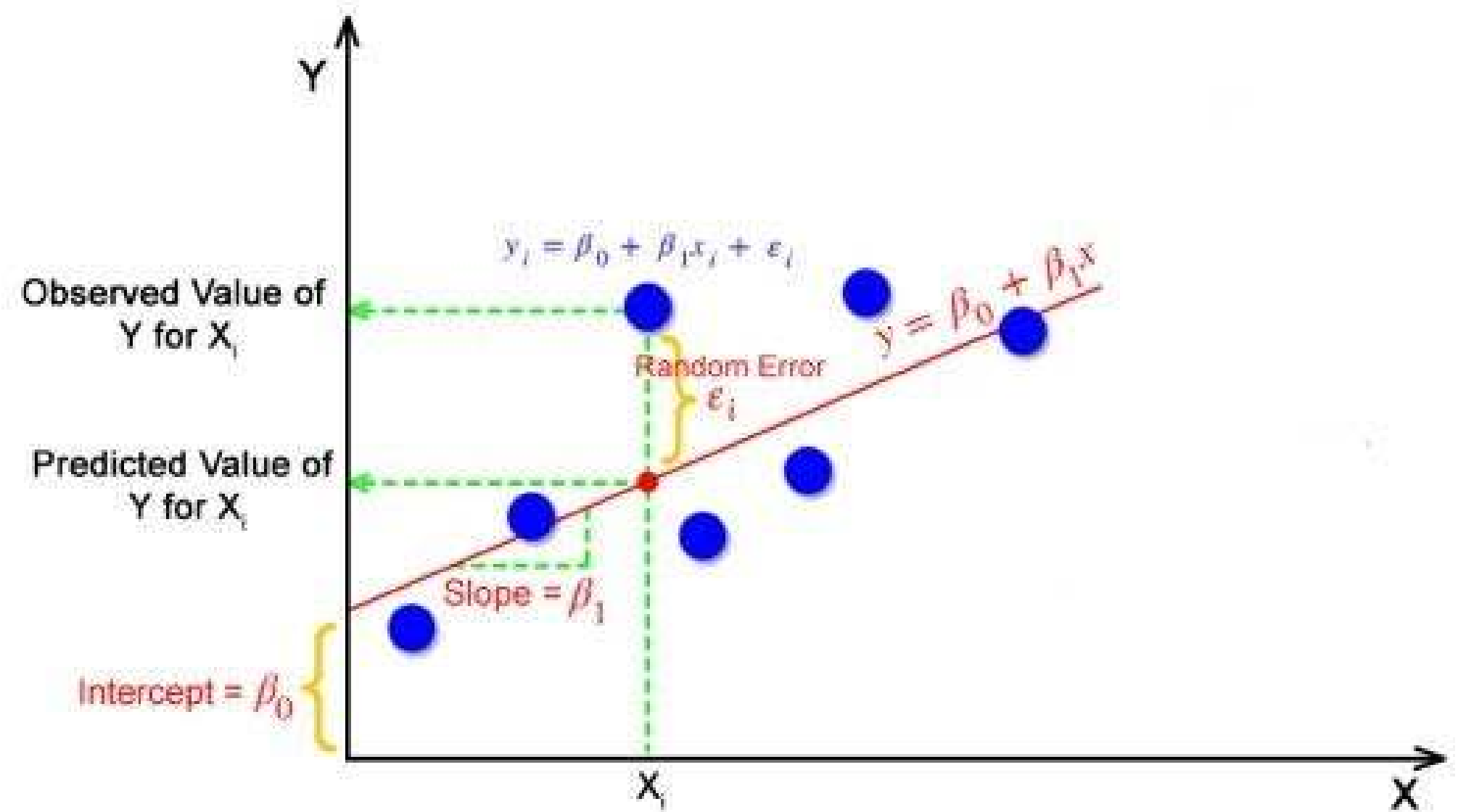
- **Simple and Interpretable:** Linear regression is easy to understand and interpret, making it a popular choice for modeling relationships between variables.
- **Predicts Relationships:** It predicts the value of a dependent variable based on one or more independent variables by finding the best-fitting straight line.
- **Assumes Linearity:** It assumes a linear relationship between the dependent and independent variables, meaning the change in the dependent variable is proportional to the change in the independent variables.
- **Sensitive to Outliers:** Linear regression can be affected by outliers, as they can significantly influence the position of the best-fitting line.

SIMPLE LINEAR REGRESSION

$$y = \beta_0 + \beta_1 X$$

where:

- Y is the dependent variable
- X is the independent variable
- β_0 is the intercept
- β_1 is the slope ($\tan \theta$)

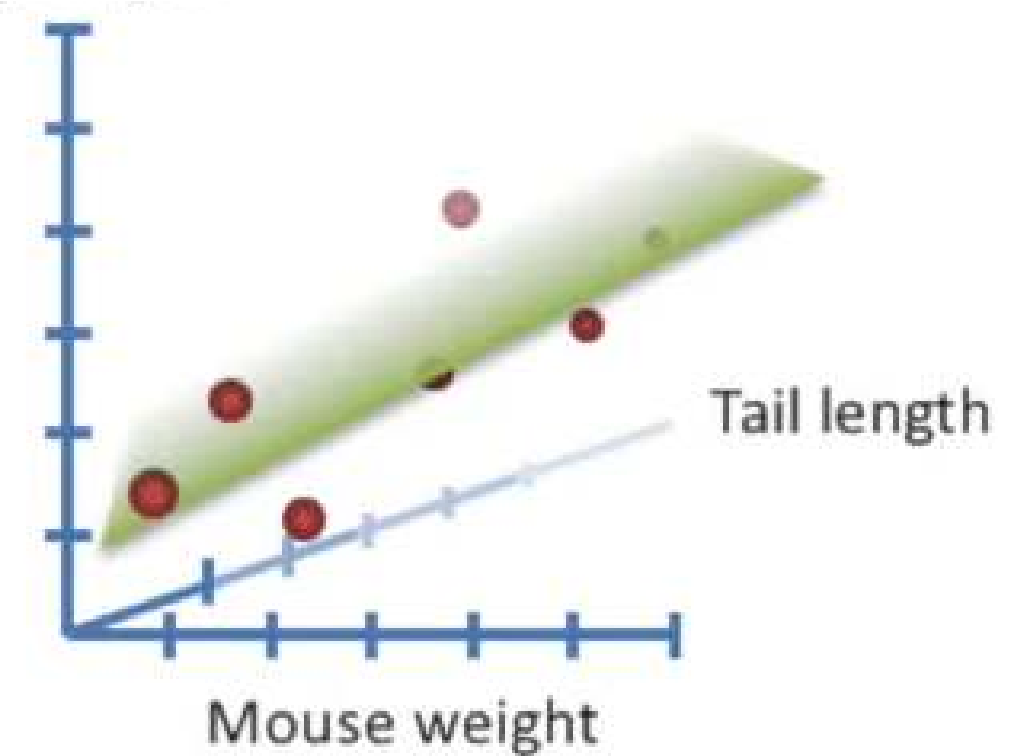


MULTIPLE LINEAR REGRESSION

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

where:

- Y is the dependent variable
- X_1, X_2, \dots, X_n are the independent variables
- β_0 is the intercept
- $\beta_1, \beta_2, \dots, \beta_n$ are the slopes



```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x_train,y_train)
```

LinearRegression ⓘ ?

LinearRegression()

```
y_pred = lm.predict(x_test)
y_pred
new_data = pd.DataFrame({
    'Ratings': [4],
    'RAM': [6],
    'ROM': [64],
    'Mobile_Size': [6.5],
    'Primary_Cam': [16],
    'Selfi_Cam': [12],
    'Battery_Power': [4000]
})
lm.predict(new_data)
```

```
array([35194.22001009])
```

lm = LinearRegression():

Object will be used to fit the model and make predictions.

lm.fit(x_train, y_train):

It trains this model using our training data, so it can learn the relationship between the features (x_train) and the target values (y_train).

y_pred : displays the predicted values (y_pred) for the test data.

lm.predict(x_test) uses the trained linear regression model (lm) to make predictions based on the test data (x_test).

lm.predict(new_data)

This uses the trained model (lm) to predict the target value based on the features provided in new_data.

Mean Squared Error

- Computes the difference between the true target values (**y_test**) and the predicted values (**y_pred**).
- Lower values indicate a better fit of the model.

Root Mean Squared Error

- By setting `squared=False`, `mean_squared_error` computes the Root Mean Squared Error (RMSE) instead of MSE.

R² (R-squared)

- It tells you how well your model's predictions match the actual data.
- It measures how much of the variation in the outcome can be explained by the inputs.
- **0 < R² < 1** - higher values indicate a better fit.

```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
print("MSE:", mse)

rmse = mean_squared_error(y_test, y_pred, squared=False)
print("RMSE:", rmse)

r2 = r2_score(y_test, y_pred)
print("R^2 Score:", r2)
```

MSE: 160721085.1679886
RMSE: 12677.581992161935
R^2 Score: 0.5270619343375811

CONCLUSION

Key Features: The model effectively used features like RAM, storage, camera quality, and battery power to predict mobile prices.

Model Accuracy: The linear regression model showed good accuracy with solid R^2 scores and low RMSE values.

Residual Insights: While generally accurate, the model struggled with outliers, suggesting room for improvement.

Practical Use: This model can help consumers and retailers estimate prices based on mobile specs.

Future Improvements: Incorporating advanced models and more data could enhance prediction accuracy.