```
# Chaitanya Mangla AI - DS B1
# Used Bike Price Prediction
# Use Regression to estimate the continuous sale price of a used bicycle based on its features
```

[51] ✓ 0.0s

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

[52] ✓ 0.0s

```
# Generate Synthetic Data
np.random.seed(42)
n_samples = 1000

brands = ['Yamaha', 'Honda', 'Suzuki', 'Bajaj', 'Royal Enfield', 'KTM']
brand = np.random.choice(brands, n_samples)

year = np.random.randint(2005, 2024, n_samples)
mileage = np.random.uniform(20, 70, n_samples)  # km/l
engine_cc = np.random.randint(100, 1000, n_samples)
power = engine_cc * np.random.uniform(0.05, 0.12, n_samples)  # rough proxy
owners = np.random.randint(1, 4, n_samples)

# price Formuala

price =(
    20000
    + (2025 - year) * -1500  # depreciation per year
    + mileage * 500
```

[53]

```python
df = pd.DataFrame({
    'brand': brand,
    'year': year,
    'mileage': mileage,
    'engine_cc': engine_cc,
    'power': power,
    'owners': owners,
    'price': price
})
df
```

✓ 0.0s

[53]

	brand	year	mileage	engine_cc	power	owners	price
0	Bajaj	2010	52.128798	598	68.642360	3	28118.321901
1	Royal Enfield	2010	57.967216	853	46.730229	1	126115.556733
2	Suzuki	2020	45.022265	564	45.191195	3	60678.958243
3	Royal Enfield	2008	47.118183	790	77.977601	1	128637.017310
4	Royal Enfield	2015	67.573928	830	68.111090	2	97272.330668
...	...	...	...	...	...	...	...
995	KTM	2010	46.122207	253	14.412969	1	34674.192182
996	Royal Enfield	2015	48.135585	538	49.616217	1	26803.021529
997	Yamaha	2007	42.061186	394	34.140903	1	34342.585182
998	Yamaha	2012	50.287765	318	21.544782	3	20452.112349
999	Royal Enfield	2018	65.936736	768	63.718275	1	78899.259450

1000 rows × 7 columns

```
df = pd.get_dummies(df, columns=['brand'], drop_first=True)
df
```

[54] ✓ 0.0s Pytho

	year	mileage	engine_cc	power	owners	price	brand_Honda	brand_KTM	brand_Royal Enfield	brand_Suzuki	brand_Yamaha
0	2010	52.128798	598	68.642360	3	28118.321901	False	False	False	False	False
1	2010	57.967216	853	46.730229	1	126115.556733	False	False	True	False	False
2	2020	45.022265	564	45.191195	3	60678.958243	False	False	False	True	False
3	2008	47.118183	790	77.977601	1	128637.017310	False	False	True	False	False
4	2015	67.573928	830	68.111090	2	97272.330668	False	False	True	False	False
...	...	...	...	...	...	...	...	...	...	...	...
995	2010	46.122207	253	14.412969	1	34674.192182	False	True	False	False	False
996	2015	48.135585	538	49.616217	1	26803.021529	False	False	True	False	False
997	2007	42.061186	394	34.140903	1	34342.585182	False	False	False	False	True
998	2012	50.287765	318	21.544782	3	20452.112349	False	False	False	False	True
999	2018	65.936736	768	63.718275	1	78899.259450	False	False	True	False	False

1000 rows × 11 columns

```
# Now comes the splitting of the features
X = df.drop(columns=['price'])
y = df['price']
```

[55] ✓ 0.0s Pytho

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state = 42)
```

[58] ✓ 0.0s

```
X_train
```

[59] ✓ 0.0s

	year	mileage	engine_cc	power	owners	brand_Honda	brand_KTM	brand_Royal Enfield	brand_Suzuki	brand_Yamaha
29	2008	50.867588	112	8.623210	2	False	False	False	True	False
535	2013	29.559535	253	21.733407	2	False	False	False	True	False
695	2007	52.538427	996	57.264026	2	False	False	False	False	True
557	2009	27.047981	594	43.732432	3	True	False	False	False	False
836	2012	22.027937	175	14.187822	3	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...
106	2012	38.347804	165	13.673020	1	False	False	False	True	False
270	2018	32.173898	972	49.595307	3	False	False	False	True	False
860	2005	57.049063	369	39.641625	3	False	False	False	False	False
435	2022	31.271938	540	53.107695	1	False	False	False	False	False
102	2005	33.244772	182	16.729290	3	False	False	False	False	True

800 rows × 10 columns

```python
# importing the Linear Regression Model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

[63] ✓ 0.0s

```
LinearRegression
▶ Parameters
```

```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
# Preprocessing: One-hot encode brand
preprocessor = ColumnTransformer([
    ('brand', OneHotEncoder(drop='first'), ['brand'])
], remainder='passthrough')

# Create pipeline
model = Pipeline([
    ('preprocess', preprocessor),
    ('regressor', LinearRegression())
])
```

[64] ✓ 0.0s

```python
y_pred = model.predict(X_test)
```

[ ]

```
<seaborn.axisgrid.FacetGrid at 0x2005cc66fd0>
```