



UMID30052539956

Project Title	Used Bike Prices - Feature Engineering and EDA
Tools	Visual Studio code / jupyter notebook
Domain	Finance Analyst
Project Difficulties level	Advance

About this file

Dataset Information

model_name

The name of the bike's model. It contains some additional information like model year,engine etc.

model_year

The year in which the model was built.

kms_driven

Total kilometers the bike has been driven.

owner

The represents which type of owner the bike has like it is first owner which means the current owner had bought this bike as new, second owner means the bike has been sold to this owner from first owner and so on.

location

The location of the seller.

mileage

Average mileage the bike gives. Its is represented as kilometer per liter of petrol (kmpl).

power

Power is in terms of Bhp. BHP is the rate at which the torque generated by the engine in a bike is delivered to the wheels. Such that faster the deliverability, higher is the speed of the motorcycle and vice versa. For a bike that consists of a lower BHP can pull higher loads and for a bike that contains a greater BHP can propel the bike at faster speeds.

Thus, BHP depends on several factors such as deliverability, weight and power generation of the bike. It's the most common way of rating a bike and determining which ones might be faster

price

This is the target variable of the dataset. t is in Indian rupee

Example: You can get the basic idea how you can create a project from here

Creating a major Machine Learning (ML) project on "Bike Prices - Feature Engineering and Exploratory Data Analysis (EDA)" requires an organized and systematic approach, especially to meet the level of detail expected for a 5-year experienced professional. Below is an outline and sample step-by-step guide that can help you build a comprehensive project.

Due to limitations here, I will provide a condensed version of the key steps involved. Let's break down the project into logical sections with major tasks:

Project Outline: Bike Prices Analysis

Dataset Columns:

- `model_name`: Name of the bike model
- `model_year`: Year the model was manufactured
- `kms_driven`: Kilometers driven by the bike
- `owner`: Owner category (first, second, etc.)
- `location`: City or region of the sale
- `mileage`: Fuel efficiency of the bike
- `power`: Power rating of the bike
- `price`: Selling price of the bike (target variable)
- `cc`: Engine capacity in cubic centimeters
- `brand`: Brand of the bike

Step-by-Step Project Guide

1. Data Collection and Preparation

1. Loading and Inspecting the Data

- Load the dataset.
- Inspect the dataset for basic structure, types, and any initial inconsistencies.

python

code

```
import pandas as pd

# Load the dataset
df = pd.read_csv("bike_prices.csv")

# Check first few rows
print(df.head())

print(df.info())
```

2.

3. Data Cleaning

- **Missing Values:** Identify and handle missing values.
- **Duplicates:** Check and remove any duplicate rows.
- **Inconsistent Data:** Address inconsistencies like typos or irregular formats in `location` or `owner`.

python

code

```
# Check missing values
print(df.isnull().sum())
```

```
# Fill missing values (if any) or decide on removal based on analysis
```

```
df = df.dropna() # For simplicity, remove rows with missing values
```

4.

2. Exploratory Data Analysis (EDA)

1. Descriptive Statistics

- Get a statistical summary of each numerical column (`price`, `kms_driven`, `mileage`, `power`, etc.).

python

code

```
print(df.describe())
```

2.

3. Univariate Analysis

- Plot histograms for continuous variables like `price`, `mileage`, `power`, `kms_driven`.
- Analyze the frequency distribution of categorical features such as `model_year`, `owner`, and `brand`.

python

code

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
# Price Distribution
```

```
plt.figure(figsize=(8,6))  
sns.histplot(df['price'], kde=True)  
plt.title('Price Distribution')  
plt.show()
```

4.

5. Bivariate Analysis

- Analyze relationships, e.g., `price` vs. `model_year`, `price` vs. `kms_driven`.
- Correlation heatmap for numerical variables to understand dependencies.

python

code

```
# Price vs Year  
plt.figure(figsize=(8,6))  
sns.scatterplot(data=df, x='model_year', y='price')  
plt.title('Price vs Model Year')  
plt.show()
```

6.

3. Feature Engineering

1. Creating New Features

- **Age of Bike:** Calculate based on `model_year` and the current year.
- **Power-to-Weight Ratio:** For bikes where both `power` and `cc` are available, create a ratio feature.
- **Location Encoding:** Use encoding techniques to convert `location` to a numerical format.

python

code

```
# Example: Age of Bike
current_year = 2023
df['bike_age'] = current_year - df['model_year']
```

2.

3. Handling Categorical Variables

- Use one-hot encoding for categorical features like `brand`, `location`, and `owner`.

python

code

```
# One-hot encode the brand and owner columns
df = pd.get_dummies(df, columns=['brand', 'owner'],
drop_first=True)
```

4.

5. Feature Transformation

- **Normalization/Standardization:** Standardize numerical features such as `price`, `kms_driven`, `mileage`, and `power`.
- **Log Transformation:** For skewed data, apply log transformation.

python

code

```
# Standardizing numerical features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['price', 'kms_driven', 'mileage', 'power']] =
```

```
scaler.fit_transform(df[['price', 'kms_driven', 'mileage',  
'power']])
```

6.

4. Data Visualization

1. Multivariate Plots

- Pairplot or scatter matrix for all numerical columns to visualize relationships.
- Boxplots for categorical columns like `brand` and `owner` against `price`.

python

code

```
# Pairplot for selected features  
sns.pairplot(df[['price', 'kms_driven', 'mileage', 'power',  
'bike_age']])  
plt.show()
```

2.

3. Geographical Analysis

- Use maps or bar plots to show average bike prices by `location`.

4. Time Series Analysis

- Analyze price trends over years if data has a temporal component.

5. Model Building

1. Splitting Data

- Separate features (X) and target variable (y, which is `price`).
- Split the data into training and testing sets.

python

code

```
from sklearn.model_selection import train_test_split

X = df.drop('price', axis=1)
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

2.

3. Model Selection

- Start with simple regression models like Linear Regression, Ridge, or Lasso.
- Experiment with tree-based models like Random Forest or Gradient Boosting.

4. Model Training and Evaluation

- Train the models and use metrics like MAE, MSE, and R^2 to evaluate performance.

python

code

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

model = LinearRegression()
model.fit(X_train, y_train)

# Predict and evaluate
predictions = model.predict(X_test)
```

```
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print(f'MSE: {mse}, R²: {r2}')
```

5.

6. Hyperparameter Tuning

- Use techniques like GridSearchCV for hyperparameter tuning to improve model performance.

python

code

```
from sklearn.model_selection import GridSearchCV

# Example: RandomForest with GridSearchCV
from sklearn.ensemble import RandomForestRegressor

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30]
}

grid_search = GridSearchCV(RandomForestRegressor(), param_grid,
cv=5)

grid_search.fit(X_train, y_train)
```

7.

6. Conclusion and Recommendations

- **Insights:** Summarize findings such as which factors most influence bike prices.
- **Model Evaluation:** Discuss model performance, strengths, and limitations.

- **Next Steps:** Suggest improvements like additional data collection, feature engineering, or advanced modeling.

Documentation

- Ensure that the report includes code explanations, visualizations, and key findings. For a lengthy document, include an index, and clearly divide sections.

Example: You can get the basic idea how you can create a project from here

Sample code with output

Used Bike Prices - EDA and Modelling

Used Bike Price prediction using Machine Learning.



In [1]:

```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
```

Data Loading

In [2]:

```
data =
pd.read_csv("../input/used-bike-price-in-india/bikes.csv")
data.head()
```

Out[2]:

	model_name	model_year	kms_driven	owner	location	mileage	power	price
0	Bajaj Avenger Cruise 220 2017	2017	17000 Km	first owner	hyderabad	\n\n 35 kmpl	19 bhp	63500
1	Royal Enfield Classic 350cc 2016	2016	50000 Km	first owner	hyderabad	\n\n 35 kmpl	19.80 bhp	115000
2	Hyosung GT250R 2012	2012	14795 Km	first owner	hyderabad	\n\n 30 kmpl	28 bhp	300000
3	Bajaj Dominar 400 ABS 2017	2017	Mileage 28 Kms	first owner	pondicherry	\n\n 28 Kms	34.50 bhp	100000
4	Jawa Perak 330cc 2020	2020	2000 Km	first owner	bangalore	\n\n	30 bhp	197500

Data Engineering

In [3]:

```
data.isna().sum()
```

Out[3]:

```
model_name      0
model_year      0
kms_driven      0
owner           0
location        19
mileage         11
power           31
price           0
```

```
dtype: int64
```

CC from Model name

From the model name, following can be extracted :

- CC of the bike
- Brand and Model name

In [4]:

```
cc = []
```

```
for veh in data.model_name:
```

```
    models = veh.split(" ")
```

```
    models = " ".join(models[:len(models)-1]).lower()
```

```
    if re.search('[0-9]*cc',models,flags=re.I) != None:
```

```

        if
models[re.search('[0-9]*cc',models,flags=re.I).start():re.searc
h('[0-9]*cc',models,flags=re.I).end()] not in ['cc']:

cc.append(models[re.search('[0-9]*cc',models,flags=re.I).start(
):re.search('[0-9]*cc',models,flags=re.I).end()])

        else:
            cc.append(models)

        elif re.search('[0-9]* (cc)',models,flags=re.I) !=
None:

            cc.append(models[re.search('[0-9]*
(cc)',models,flags=re.I).start():re.search('[0-9]*
(cc)',models,flags=re.I).end()])

        elif "hyosung" in models:
            cc.append(models)

        else:
            cc.append(models)

```

Few entries require manual processing because of their different naming convention and non availability of engine cc information in their naming

```

In [5]:
for i in range(len(cc)):
    if "1000" in cc[i]:
        cc[i] = "1000cc"

```

```
elif "310" in cc[i]:  
    cc[i] = "310cc"  
elif "apache rtr 200" in cc[i]:  
    cc[i] = "200cc"  
elif "ns200" in cc[i]:  
    cc[i] = "200cc"  
elif "rs200" in cc[i]:  
    cc[i] = "200cc"  
elif "220" in cc[i]:  
    cc[i] = "220cc"  
elif "400" in cc[i]:  
    cc[i] = "400cc"  
elif "250" in cc[i]:  
    cc[i] = "250cc"  
elif "125" in cc[i]:  
    cc[i] = "125cc"  
elif "160" in cc[i]:  
    cc[i] = "160cc"  
elif "150" in cc[i]:  
    cc[i] = "150cc"  
elif "350" in cc[i]:  
    cc[i] = "350cc"  
elif "200" in cc[i]:  
    cc[i] = "200cc"  
elif "100" in cc[i]:
```



```
        cc[i] = "100cc"
elif "180" in cc[i]:
        cc[i] = "180cc"
elif "110" in cc[i]:
        cc[i] = "110cc"
elif "390" in cc[i]:
        cc[i] = "390cc"
elif "135" in cc[i]:
        cc[i] = "135cc"
elif "r15" in cc[i]:
        cc[i] = "150cc"
elif "650" in cc[i]:
        cc[i] = "650cc"
elif "750" in cc[i]:
        cc[i] = "750cc"
elif "800" in cc[i]:
        cc[i] = "800cc"
elif "300" in cc[i]:
        cc[i] = "300cc"
elif "765" in cc[i]:
        cc[i] = "765cc"
elif "883" in cc[i]:
        cc[i] = "883cc"
elif "797" in cc[i]:
        cc[i] = "797cc"
```

```
elif "810" in cc[i]:
    cc[i] = "810cc"
elif "321" in cc[i]:
    cc[i] = "321cc"
elif "821" in cc[i]:
    cc[i] = "821cc"
elif "120" in cc[i]:
    cc[i] = "120cc"
elif "1745" in cc[i]:
    cc[i] = "1745cc"
elif "899" in cc[i]:
    cc[i] = "899cc"
elif "900" in cc[i]:
    cc[i] = "900cc"
elif "302" in cc[i]:
    cc[i] = "302cc"
elif "959" in cc[i]:
    cc[i] = "959cc"
elif "600" in cc[i]:
    cc[i] = "600cc"
elif "502" in cc[i]:
    cc[i] = "502cc"
elif "um renegade" in cc[i]:
    cc[i] = "279cc"
elif "hero splendor" in cc[i]:
```

```
        cc[i] = "97cc"
elif "hero passion plus" in cc[i]:
    cc[i] = "97cc"
elif "yamaha fz" in cc[i]:
    cc[i] = "150cc"
elif "honda hornet" in cc[i]:
    cc[i] = "184cc"
elif "royal enfield interceptor" in cc[i]:
    cc[i] = "650cc"
elif "hero passion pro" in cc[i]:
    cc[i] = "113cc"
elif "hero passion xpro" in cc[i]:
    cc[i] = "109cc"
elif "harley-davidson street bob" in cc[i]:
    cc[i] = "1868cc"
elif "harley-davidson fat bob" in cc[i]:
    cc[i] = "1868cc"
elif "harley-davidson fat boy" in cc[i]:
    cc[i] = "1868cc"
elif "harley-davidson street rod" in cc[i]:
    cc[i] = "749cc"
elif "zx-10r" in cc[i]:
    cc[i] = "1000cc"
elif "rsv4" in cc[i]:
    cc[i] = "1099cc"
```

```
elif "tvS sport" in cc[i]:
    cc[i] = "109cc"
elif "tvS star city" in cc[i]:
    cc[i] = "109cc"
elif "harley-davidson superlow" in cc[i]:
    cc[i] = "883cc"
elif "harley-davidson roadster" in cc[i]:
    cc[i] = "1202cc"
elif "harley-davidson forty eight" in cc[i]:
    cc[i] = "1202cc"
elif "harley-davidson night rod special" in cc[i]:
    cc[i] = "1247cc"
elif "triumph rocket iii roadster" in cc[i]:
    cc[i] = "2458cc"
elif "triumph thunderbird lt" in cc[i]:
    cc[i] = "1699cc"
elif "kawasaki vulcan s black" in cc[i]:
    cc[i] = "649cc"
elif "mahindra mojo black pearl" in cc[i]:
    cc[i] = "300cc"
elif "ducati diavel carbon" in cc[i]:
    cc[i] = "1198cc"
elif "triumph tiger explorer" in cc[i]:
    cc[i] = "1215cc"
elif "royal enfield continental" in cc[i]:
```


[illegible]

78 52	Yamaha YZF-R15 150cc 2011	2011	7000 Km	first owne r	agra	\n\n 42 kmpl	16 bhp	550 00	1 5 0
78 53	Bajaj Discover 100cc 2015	2015	Mileage 80 Kmpl	first owne r	delhi	\n\n 80 Kmpl	7.7	280 00	1 0 0
78 54	Bajaj Pulsar 180cc 2016	2016	6407 Km	first owne r	banga lore	\n\n 65 kmpl	17 bhp	617 40	1 8 0
78 55	Bajaj V15 150cc 2016	2016	7524 Km	first owne r	banga lore	\n\n 57 kmpl	11.8 0 bhp	490 00	1 5 0
78 56	Bajaj Pulsar 220cc 2016	2016	15000 Km	first owne r	chenn ai	\n\n 38 kmpl	21 bhp	650 00	2 2 0

7857 rows x 9 columns

Split the model name by spaces and take first split as the brand name

In [8]:

```
np.unique([i.split()[0] for i in data.model_name])
```

Out[8]:

```
array(['Aprilia', 'BMW', 'Bajaj', 'Benelli',  
      'BenelliImperiale', 'Ducati',  
      'Fb', 'Harley-Davidson', 'Hero', 'Honda', 'Husqvarna',  
      'Hyosung',  
      'Ideal', 'Indian', 'Jawa', 'KTM', 'Kawasaki', 'MV',  
      'Mahindra',  
      'Moto', 'Royal', 'Suzuki', 'TVS', 'Triumph', 'UM',  
      'Yamaha',  
      'Yazdi', 'yamaha'], dtype='<U16')
```

Benelli has inconsistent naming convention shows repeating values. Preprocess them to generate spacing between brand and model name

In [9]:

```
data[data.model_name.str.contains("BenelliImperiale") > 0]
```

Out[9]:

	model_name	model_year	kms_driven	owner	location	mileage	power	price	cc
2809	BenelliImperiale 400 BS6 2020	2020	2800 Km	first owner	mumbai	\n\n35 kmpl	20.7 bhp	207500	400
2978	BenelliImperiale 400 2020	2020	1500 Km	first owner	mumbai	\n\n	20.7 bhp	207500	400
2979	BenelliImperiale 400 2020	2020	3500 Km	first owner	mumbai	\n\n	20.7 bhp	206462	400
4177	BenelliImperiale 400 2020	2020	3500 Km	first owner	mumbai	\n\n	20.7 bhp	220000	400
4399	BenelliImperiale 400 2020	2020	7700 Km	first owner	bangalore	\n\n	20.7 bhp	220000	400

				r					0
51 77	BenelliImperiale 400 2020	2020	1900 Km	first owne r	secund erabad	\n\n	20.7 bhp	240 000	4 0 0

In [10]:

```
data.model_name.replace('BenelliImperiale 400 2020', 'Benelli  
Imperiale 400 2020', inplace=True)  
data.model_name.replace('BenelliImperiale 400 BS6  
2020', 'Benelli Imperiale 400 BS6 2020', inplace=True)
```

In [11]:

```
data[data.model_name.str.contains("Benelli") > 0]
```

Out[11]:

	model_name	model _year	kms_ driven	owner	location	milea ge	powe r	pric e	c c
16	Benelli 302R	2017	11000	first	bangalo	\n\n	38.26	230	3 0

51	300CC 2017		Km	owner	re		bhp	000	0
16 68	Benelli 302R 300CC 2018	2018	23933 Km	first owner	bangalo re	\n\n	38.26 bhp	270 000	3 0 0
19 26	Benelli 302R 300CC 2018	2018	15025 Km	secon d owner	mumbai	\n\n	38.26 bhp	240 000	3 0 0
19 39	Benelli Leoncino 500cc 2019	2019	3300 Km	first owner	delhi	\n\n	46.8 bhp	435 262	5 0 0
21 40	Benelli TNT 300 ABS 2020	2020	4500 Km	first owner	delhi	\n\n 25 kmpl	37.73 bhp	295 000	3 0 0
23 07	Benelli TNT 300 2017	2017	10000 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	259 375	3 0 0

24 20	Benelli TNT 300 ABS 2020	2020	13000 Km	first owner	belgau m	\n\n 25 kmpl	37.73 bhp	315 000	3 0 0
24 35	Benelli TNT 899 2015	2015	3174 Km	first owner	jajpur	\n\n 17 kmpl	119.3 7 bhp	569 400	8 9 9
25 54	Benelli 302R 300CC 2019	2019	12000 Km	first owner	ranchi	\n\n	38.26 bhp	269 000	3 0 0
26 52	Benelli TNT 600 GT 2016	2016	11000 Km	first owner	chandig arh	\n\n 19 kmpl	82.70 bhp	497 000	6 0 0
28 09	Benelli Imperiale 400 BS6 2020	2020	2800 Km	first owner	mumbai	\n\n 35 kmpl	20.7 bhp	207 500	4 0 0
29	Benelli TNT 300	2017	11000	first	mumbai	\n\n 25	37.73	240	3 0

32	2017		Km	owner		kmpl	bhp	000	0
29 78	Benelli Imperiale 400 2020	2020	1500 Km	first owner	mumbai	\n\n	20.7 bhp	207 500	4 0 0
29 79	Benelli Imperiale 400 2020	2020	3500 Km	first owner	mumbai	\n\n	20.7 bhp	206 462	4 0 0
30 20	Benelli TNT 300 2016	2016	6000 Km	first owner	delhi	\n\n 25 kmpl	37.73 bhp	255 000	3 0 0
32 81	Benelli TNT 899 2017	2017	5000 Km	secon d owner	bangalo re	\n\n 17 kmpl	119.3 7 bhp	600 000	8 9 9
33 66	Benelli 302R 300CC 2017	2017	13236 Km	first owner	chennai	\n\n	38.26 bhp	218 000	3 0 0

33 90	Benelli 302R 300CC 2017	2017	17000 Km	first owner	chennai	\n\n	38.26 bhp	250 000	3 0 0
35 88	Benelli 302R 2017	2017	12000 Km	first owner	jalandh ar	\n\n	38.26 bhp	250 000	3 0 2
36 47	Benelli TNT 300 ABS 2019	2019	5000 Km	first owner	chennai	\n\n 25 kmpl	37.73 bhp	295 000	3 0 0
36 75	Benelli TNT 300 2016	2016	17500 Km	secon d owner	pune	\n\n 25 kmpl	37.73 bhp	151 045	3 0 0
37 59	Benelli TNT 300 2016	2016	30000 Km	secon d owner	mhasla	\n\n 25 kmpl	37.73 bhp	185 000	3 0 0
37	Benelli TNT 25	2017	33500	first	chennai	\n\n 25	28.16	125	2 5

86	250cc 2017		Km	owner		kmpl	bhp	000	0
38 99	Benelli TNT 300 2016	2016	9000 Km	first owner	chennai	\n\n 25 kmpl	37.73 bhp	203 921	3 0 0
39 08	Benelli Leoncino 250 2019	2019	11000 Km	first owner	secund erabad	\n\n	25.1 bhp	208 800	2 5 0
39 20	Benelli 302R 300CC 2019	2019	29300 Km	first owner	ahmeda bad	\n\n	38.26 bhp	280 000	3 0 0
39 72	Benelli TNT 300 2015	2015	21500 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	180 000	3 0 0
39 77	Benelli 302S 2021	2021	16000 Km	first owner	ahmeda bad	\n\n	NaN	315 000	3 0 2

41 18	Benelli Leoncino 500cc 2020	2020	1000 Km	first owner	mumbai	\n\n	46.8 bhp	466 875	5 0 0
41 33	Benelli TNT 300 2016	2016	15000 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	228 250	3 0 0
41 35	Benelli TNT 300 2018	2018	10000 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	259 375	3 0 0
41 37	Benelli TNT 300 2016	2016	3500 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	233 437	3 0 0
41 42	Benelli TNT 600i 2016	2016	14000 Km	first owner	mumbai	\n\n 19 kmpl	82.70 bhp	415 000	6 0 0
41	Benelli Imperiale 400	2020	3500	first	mumbai	\n\n	20.7	220	4 0

77	2020		Km	owner			bhp	000	0
43 99	Benelli Imperiale 400 2020	2020	7700 Km	first owner	bangalo re	\n\n	20.7 bhp	220 000	4 0 0
44 37	Benelli TNT 600 GT 2018	2018	26500 Km	first owner	hyderab ad	\n\n 19 kmpl	82.70 bhp	515 000	6 0 0
45 46	Benelli TNT 25 250cc 2016	2016	25000 Km	secon d owner	pune	\n\n 25 kmpl	28.16 bhp	106 000	2 5 0
46 30	Benelli 302R 300CC 2019	2019	2500 Km	first owner	bangalo re	\n\n	38.26 bhp	310 000	3 0 0
47 00	Benelli TRK 502 2019	2019	3250 Km	first owner	bangalo re	\n\n	47 bhp	500 000	5 0 2

47 48	Benelli TNT 300 2017	2017	10000 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	230 000	3 0 0
49 97	Benelli TNT 300 2016	2016	6400 Km	secon d owner	pune	\n\n 25 kmpl	37.73 bhp	200 000	3 0 0
49 98	Benelli TNT 300 2016	2016	32000 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	218 100	3 0 0
50 59	Benelli TNT 300 2017	2017	9500 Km	first owner	chennai	\n\n 25 kmpl	37.73 bhp	215 000	3 0 0
51 77	Benelli Imperiale 400 2020	2020	1900 Km	first owner	secund erabad	\n\n	20.7 bhp	240 000	4 0 0
53	Benelli TNT 25	2018	14000	first	chennai	\n\n 25	28.16	170	2 5

05	250cc 2018		Km	owner		kmpl	bhp	000	0
53 83	Benelli TNT 899 2018	2018	4500 Km	first owner	delhi	\n\n 17 kmpl	119.3 7 bhp	700 000	8 9 9
55 09	Benelli TNT 300 2015	2015	21000 Km	first owner	kolkata	\n\n 25 kmpl	37.73 bhp	210 000	3 0 0
56 83	Benelli TNT 300 2017	2017	11000 Km	first owner	bhuban eshwar	\n\n 25 kmpl	37.73 bhp	260 000	3 0 0
57 18	Benelli TNT 300 2017	2017	411 Km	first owner	kolkata	\n\n 25 kmpl	37.73 bhp	295 000	3 0 0
57 22	Benelli TNT 300 2017	2017	5000 Km	first owner	delhi	\n\n 25 kmpl	37.73 bhp	275 000	3 0 0

57 28	Benelli TNT 300 ABS 2019	2019	10700 Km	first owner	hyderab ad	\n\n 25 kmpl	37.73 bhp	280 000	3 0 0
65 94	Benelli TNT 600i ABS 2018	2018	11500 Km	secon d owner	mumbai	\n\n 19 kmpl	82.70 bhp	475 000	6 0 0
66 36	Benelli TNT 600 GT 2018	2018	10800 Km	first owner	ahmeda bad	\n\n 19 kmpl	82.70 bhp	425 000	6 0 0
67 77	Benelli TNT 300 i 300cc 2016	2016	7500 Km	first owner	mumbai	\n\n 25 kmpl	37.73 bhp	270 000	3 0 0
67 81	Benelli TNT 25 250 cc 2016	2016	6300 Km	first owner	lonaval a	\n\n	NaN	140 000	2 5 0
67	Benelli TNT 300	2017	11700	secon d	mumbai	\n\n 25	37.73	280	3 0

83	2017		Km	owner		kmpl	bhp	000	0
67 88	Benelli TNT 300 2016	2016	13000 Km	secon d owner	bardha man	\n\n 25 kmpl	37.73 bhp	220 000	3 0 0
67 91	Benelli TNT 600i ABS 2019	2019	1000 Km	first owner	kolkata	\n\n 19 kmpl	82.70 bhp	650 000	6 0 0
70 23	Benelli 302R 2017	2017	4130 Km	first owner	mumbai	\n\n	38.26 bhp	300 000	3 0 2

Extracing Brand from model name

In [12]:

```
brands = [i.split()[0] for i in data.model_name]
data['brand'] = brands
```

In [13]:

data

Out[13]:

	model_name	model_year	kms_driven	owner	location	mileage	power	price	cc	brand
0	Bajaj Avenger Cruise 220 2017	2017	17000 Km	first owner	hyderabad	\n\n35 kmpl	19 bhp	63500	220	Bajaj
1	Royal Enfield Classic 350cc 2016	2016	50000 Km	first owner	hyderabad	\n\n35 kmpl	19.80 bhp	115000	350	Royal
2	Hyosung GT250R 2012	2012	14795 Km	first owner	hyderabad	\n\n30 kmpl	28 bhp	300000	250	Hyosung
3	Bajaj Dominar 400 ABS 2017	2017	Mileage 28 Kms	first owner	pondicherry	\n\n28 Kms	34.50 bhp	100000	400	Bajaj

4	Jawa Perak 330cc 2020	2020	2000 Km	first own er	banga lore	\n\n	30 bhp	197 500	3 3 0	Jaw a
...
78 52	Yamaha YZF-R15 150cc 2011	2011	7000 Km	first own er	agra	\n\n 42 kmpl	16 bhp	550 00	1 5 0	Yam aha
78 53	Bajaj Discover 100cc 2015	2015	Mileage 80 Kmpl	first own er	delhi	\n\n 80 Kmpl	7.7	280 00	1 0 0	Baja j
78 54	Bajaj Pulsar 180cc 2016	2016	6407 Km	first own er	banga lore	\n\n 65 kmpl	17 bhp	617 40	1 8 0	Baja j
78 55	Bajaj V15 150cc 2016	2016	7524 Km	first own er	banga lore	\n\n 57 kmpl	11.8 0 bhp	490 00	1 5 0	Baja j

78 56	Bajaj Pulsar 220cc 2016	2016	15000 Km	first own er	chenn ai	\n\n 38 kmpl	21 bhp	650 00	2 2 0	Baja j
----------	----------------------------	------	-------------	--------------------	-------------	--------------------	-----------	-----------	-------------	-----------

7857 rows x 10 columns

In [14]:

```
data.replace('Ideal', "Jawa", inplace=True)
data.replace("yamaha", "Yamaha", inplace=True)
```

Preprocessing Mileage

In [15]:

```
mil = data.mileage.to_list()
mil = [str(i).lower().replace('kmpl', '').split("-")[0] for i in mil]
mil = [str(i).lower().replace('kms', '') for i in mil]
mil = [str(i).replace('\n', '') for i in mil]
mil = [str(i).strip() for i in mil]
data.mileage = mil
```

In [16]:

```
data[data.mileage == 'liquid cooled']
```


Out[16]:

	model_name	model_year	kms_driven	owner	location	mileage	power	price	cc	brand
157	Jawa Forty Two 295CC 2019	2019	4930 Km	first owner	moradabad	liquid cooled	27 bhp	140000	295	Jawa
211	Jawa Forty Two 295CC Dual ABS BS6 2020	2020	5000 Km	first owner	pune	liquid cooled	27 bhp	190000	295	Jawa
300	Jawa Forty Two 295CC 2019	2019	600 Km	first owner	ludhiana	liquid cooled	27 bhp	162000	295	Jawa
347	Jawa Forty Two 295CC 2019	2019	3200 Km	first owner	ghazibad	liquid cooled	27 bhp	160000	295	Jawa

54 9	Jawa Forty Two Dual ABS 295CC 2020	2020	419 Km	first owne r	chenn ai	liquid coole d	27 bh p	165 000	2 9 5	Ja wa
65 6	Jawa Forty Two 295CC 2019	2019	1800 Km	first owne r	nagpu r	liquid coole d	27 bh p	142 000	2 9 5	Ja wa
66 8	Jawa Standard 295CC ABS BS6 2020	2020	3000 Km	first owne r	rajkot	liquid coole d	27 bh p	167 000	2 9 5	Ja wa
69 9	Jawa Standard 295CC Dual ABS BS6 2020	2020	2700 Km	first owne r	mumb ai	liquid coole d	27 bh p	160 000	2 9 5	Ja wa
77 5	Jawa Forty Two Dual ABS 295CC 2020	2020	1700 Km	first owne r	allaha bad	liquid coole d	27 bh p	220 000	2 9 5	Ja wa
81	Jawa Forty Two	2019	11500	first owne	madur	liquid coole	27 bh	170	2 9	Ja

5	295CC 2019		Km	r	ai	d	p	000	5	wa
10 96	Jawa Forty Two 295CC 2019	2019	10500 Km	first owne r	mumb ai	liquid coole d	27 bh p	150 000	2 9 5	Ja wa
12 19	Jawa Standard 295CC 2020	2020	1003 Km	first owne r	hyder abad	liquid coole d	27 bh p	175 000	2 9 5	Ja wa
13 07	Jawa Forty Two 295CC 2019	2019	2450 Km	first owne r	pune	liquid coole d	27 bh p	150 000	2 9 5	Ja wa
13 19	Jawa Forty Two 295CC 2019	2019	7541 Km	first owne r	delhi	liquid coole d	27 bh p	144 000	2 9 5	Ja wa
15 45	Jawa Forty Two 295CC 2019	2019	1600 Km	first owne r	mumb ai	liquid coole d	27 bh p	155 625	2 9 5	Ja wa

18 84	Jawa Forty Two 295CC 2019	2019	3800 Km	first owne r	panvel	liquid coole d	27 bh p	138 750	2 9 5	Ja wa
20 21	Jawa Standard 295CC 2019	2019	2400 Km	first owne r	delhi	liquid coole d	27 bh p	150 000	2 9 5	Ja wa
25 65	Jawa Forty Two 295CC 2019	2019	47000 Km	secon d owne r	hyder abad	liquid coole d	27 bh p	160 000	2 9 5	Ja wa
25 67	Jawa Forty Two 295CC 2019	2019	4600 Km	first owne r	hyder abad	liquid coole d	27 bh p	155 000	2 9 5	Ja wa
30 03	Jawa Forty Two 295CC Dual ABS BS6 2020	2020	1100 Km	first owne r	delhi	liquid coole d	27 bh p	165 000	2 9 5	Ja wa

30 04	Jawa Forty Two 295CC Dual ABS BS6 2021	2021	700 Km	first owne r	delhi	liquid coole d	27 bh p	180 000	2 9 5	Ja wa
30 22	Jawa Standard 295CC 2020	2020	800 Km	first owne r	delhi	liquid coole d	27 bh p	190 000	2 9 5	Ja wa
32 04	Jawa Forty Two 295CC 2019	2019	7500 Km	first owne r	jaipur	liquid coole d	27 bh p	138 300	2 9 5	Ja wa
37 25	Jawa Standard 295CC 2020	2020	8280 Km	first owne r	ajmer	liquid coole d	27 bh p	164 400	2 9 5	Ja wa
38 11	Jawa Forty Two 295CC 2019	2019	10000 Km	secon d owne r	banga lore	liquid coole d	27 bh p	150 000	2 9 5	Ja wa

38 34	Jawa Forty Two 295CC Dual ABS BS6 2020	2020	5500 Km	first owne r	mohali	liquid coole d	27 bh p	180 000	2 9 5	Ja wa
42 82	Jawa Forty Two 295CC 2020	2020	8000 Km	first owne r	banga lore	liquid coole d	27 bh p	165 000	2 9 5	Ja wa
43 47	Jawa Forty Two 295CC 2019	2019	15451 Km	first owne r	chenn ai	liquid coole d	27 bh p	139 000	2 9 5	Ja wa
43 83	Jawa Forty Two 295CC 2020	2020	1100 Km	first owne r	ahme dabad	liquid coole d	27 bh p	123 750	2 9 5	Ja wa
44 83	Jawa Standard 295CC Dual ABS BS6 2021	2021	1000 Km	first owne r	indore	liquid coole d	27 bh p	190 000	2 9 5	Ja wa
45	Jawa Forty Two	2020	9600	first owne	mumb	liquid coole	27 bh	150	2 9	Ja

68	295CC 2020		Km	r	ai	d	p	437	5	wa
49 11	Jawa Forty Two 295CC 2019	2019	2100 Km	first owne r	hosur	liquid coole d	27 bh p	210 000	2 9 5	Ja wa
50 57	Jawa Forty Two 295CC 2019	2019	2388 Km	first owne r	chenn ai	liquid coole d	27 bh p	175 000	2 9 5	Ja wa
54 06	Jawa Forty Two 295CC 2019	2019	5400 Km	first owne r	banga lore	liquid coole d	27 bh p	168 000	2 9 5	Ja wa
58 23	Jawa Forty Two 295CC 2019	2019	2000 Km	first owne r	ghazia bad	liquid coole d	27 bh p	155 000	2 9 5	Ja wa
59 94	Jawa Forty Two 295CC 2019	2019	245 Km	first owne r	farida bad	liquid coole d	27 bh p	137 800	2 9 5	Ja wa

Replacing wrong values with their correct ones.

In [17]:

```
data.replace('liquid cooled','37.6',inplace=True)
```

In [18]:

```
np.unique(data.mileage)
```

Out[18]:

```
array(['', '104', '12', '12.5', '13', '14', '15', '16', '17',  
      '17.85',  
      '18', '18.86', '19', '20', '20.3', '20.40', '21', '22',  
      '23', '25',  
      '26', '27', '28', '29', '30', '31.85', '32', '35', '37',  
      '37.6',  
      '38', '38.5', '39.1', '39.4', '40', '42', '43', '45',  
      '45.8',  
      '46.40', '48', '5', '50', '52', '53', '53.72', '54',  
      '55', '55.47',  
      '56', '57', '58', '59', '60', '62', '63', '63.97', '64',  
      '65',  
      '67', '68', '69', '70', '71', '72', '74', '75', '77',  
      '80', '81',
```



```
'82', '82.4', '83', '84', '85', '89', '90', '95',  
'96.9', 'nan'],
```

```
dtype=object)
```

```
In [19]:
```

```
data[data.mileage == 'nan']
```

```
Out[19]:
```

	model_name	model_year	kms_driven	owner	location	mileage	power	price	cc	brand
144	TVS Apache RTR 160cc 2014	2014	20372 Km	first owner	delhi	nan	15.2 bhp @ 8500 rpm	42000	160	TVS
193	Hero Karizma R 223cc 2010	2010	70000 Km	first owner	noida	nan	17 bhp @ 7,000 rpm	19215	223	Hero

42 2	TVS Apache 150cc 2007	2007	36486 Km	seco nd own er	bathin da	nan	13.7 Bhp @ 8500 rpm	195 00	1 5 0	TVS
13 76	KTM RC 390cc 2015	2015	28000 Km	seco nd own er	hyder abad	nan	43 bhp @ 9,000 rpm	130 000	3 9 0	KTM
14 88	Royal Enfield Thunderbird 350cc 2015	2015	24000 Km	first own er	bang alore	nan	19.80 bhp @ 5,250 rpm	115 000	3 5 0	Royal
16 40	Kawasaki Ninja 650cc 2020	2020	8000 Km	seco nd own er	delhi	nan	66.4 bhp @ 8,000 rpm	520 000	6 5 0	Kawas aki
21 60	Harley-Davi dson Street	2019	600 Km	first own	bang alore	nan	70 bhp	700 700	7 4	Harley -David

	Rod 2019			er					9	son
22 77	KTM Duke 200cc 2012	2012	30000 Km	first own er	bang alore	nan	25 bhp @ 10,000 rpm	850 00	2 0 0	KTM
46 46	Hero Passion Pro 100cc 2014	2014	16834 Km	first own er	chand igarh	nan	8.2 Bhp @ 8000 rpm	400 00	1 0 0	Hero
46 55	Bajaj Pulsar 180cc 2013	2013	82000 Km	first own er	chand igarh	nan	15 bhp @ 9,000 rpm	400 00	1 8 0	Bajaj
73 94	Yamaha YZF-R15 2.0 150cc 2015	2015	18000 Km	seco nd own er	delhi	nan	16.70 bhp @ 8,500 rpm	620 00	1 5 0	Yama ha

Mileage values taken from internet

In [20]:

```
data.iloc[144,data.columns.get_loc('mileage')] = "54"
data.iloc[193,data.columns.get_loc('mileage')] = "45"
data.iloc[422,data.columns.get_loc('mileage')] = "62"
data.iloc[1376,data.columns.get_loc('mileage')] = "25"
data.iloc[1488,data.columns.get_loc('mileage')] = "40"
data.iloc[1640,data.columns.get_loc('mileage')] = "25"
data.iloc[2160,data.columns.get_loc('mileage')] = "18"
data.iloc[2277,data.columns.get_loc('mileage')] = "35"
data.iloc[4646,data.columns.get_loc('mileage')] = "65"
data.iloc[4655,data.columns.get_loc('mileage')] = "45"
data.iloc[7394,data.columns.get_loc('mileage')] = "40"
```

Few entires have just an empty string as the input. Replace them to 0 and further modify it as teh average mileage of that brand bikes.

In [21]:

```
data.replace('', '0', inplace=True)
```

In [22]:

```
null_brands = np.unique(data[data.mileage == '0'].brand)
for brand in null_brands:
    values = data[data.brand == brand].mileage.to_numpy()
    values = np.delete(values, np.where(values == '0'))
```

```

if(len(values) == 0):
    print(brand)
    mean=0
else:
    mean = values.astype("float32").mean()

for index,rows in data[data.brand == brand].iterrows():
    if data.iloc[index,data.columns.get_loc("mileage")] ==
'0':
        data.iloc[index,data.columns.get_loc("mileage")] =
mean

```

BMW

Fb

Yazdi

Few values have no mileage given to taken mean, so input them manually - values taken from internet.

In [23]:

```

data.iloc[2023,data.columns.get_loc("mileage")] = "30"
data.iloc[2832,data.columns.get_loc("mileage")] = "30"
data.iloc[3400,data.columns.get_loc("mileage")] = "30"
data.iloc[3827,data.columns.get_loc("mileage")] = "30"

```

```
data.iloc[4154,data.columns.get_loc("mileage")] = "30"
data.iloc[4392,data.columns.get_loc("mileage")] = "30"
data.iloc[5435,data.columns.get_loc("mileage")] = "30"
data.iloc[5586,data.columns.get_loc("mileage")] = "17"
data.iloc[5727,data.columns.get_loc("mileage")] = "30"
data.iloc[6561,data.columns.get_loc("mileage")] = "30"
```

In [24]:

```
data.iloc[3773,data.columns.get_loc("mileage")] = "26"
data.iloc[5031,data.columns.get_loc("mileage")] = "26"
```

In [25]:

```
data.iloc[5584,data.columns.get_loc("mileage")] = "35"
```

Preprocessing Location

Replace missing location entries with mode values

In [26]:

```
for index,rows in data[data.location.isna()].iterrows():
    data.iloc[index,data.columns.get_loc("location")] =
data[data.brand == rows.brand].dropna().location.mode()
```

Preprocessing Vehicle Power

In [27]:

```
val= data.power.isna()  
np.where(val == True)
```

Out[27]:

```
(array([ 135, 1627, 1691, 1716, 2442, 2487, 2638, 2659, 2714,  
2819, 3088,  
        3977, 4393, 4425, 4591, 4760, 5626, 5673, 5694, 5788,  
5950, 6155,  
        6321, 6781, 6853, 7306, 7442, 7513, 7783, 7796,  
7801]),)
```

Few power entries does not have any value - enter them manually using values from internet.

In [28]:

```
data.iloc[135,data.columns.get_loc("power")] = "28"  
data.iloc[1627,data.columns.get_loc("power")] = "14"  
data.iloc[1691,data.columns.get_loc("power")] = "18"  
data.iloc[1716,data.columns.get_loc("power")] = "18"  
data.iloc[2442,data.columns.get_loc("power")] = "87"  
data.iloc[2487,data.columns.get_loc("power")] = "18"  
data.iloc[2638,data.columns.get_loc("power")] = "16"
```

```
data.iloc[2659,data.columns.get_loc("power")] = "64"
data.iloc[2714,data.columns.get_loc("power")] = "18"
data.iloc[2819,data.columns.get_loc("power")] = "87"
data.iloc[3088,data.columns.get_loc("power")] = "18"
data.iloc[3977,data.columns.get_loc("power")] = "37.5"
data.iloc[4393,data.columns.get_loc("power")] = "18"
data.iloc[4425,data.columns.get_loc("power")] = "111"
data.iloc[4591,data.columns.get_loc("power")] = "75.1"
data.iloc[4760,data.columns.get_loc("power")] = "18"
data.iloc[5626,data.columns.get_loc("power")] = "65"
data.iloc[5673,data.columns.get_loc("power")] = "89.2"
data.iloc[5694,data.columns.get_loc("power")] = "64"
data.iloc[5788,data.columns.get_loc("power")] = "11"
data.iloc[5950,data.columns.get_loc("power")] = "26.21"
data.iloc[6155,data.columns.get_loc("power")] = "15.2"
data.iloc[6321,data.columns.get_loc("power")] = "26.21"
data.iloc[6781,data.columns.get_loc("power")] = "28.15"
data.iloc[6853,data.columns.get_loc("power")] = "15.2"
data.iloc[7306,data.columns.get_loc("power")] = "10.70"
data.iloc[7442,data.columns.get_loc("power")] = "7.4"
data.iloc[7513,data.columns.get_loc("power")] = "23.17"
data.iloc[7783,data.columns.get_loc("power")] = "24.13"
data.iloc[7796,data.columns.get_loc("power")] = "7.4"
data.iloc[7801,data.columns.get_loc("power")] = "8.24"
```


Resolve inconsistent naming conventions in power column by removing character values like units -bhp, kw, ps etc. Similarly, few entries are in different units - Kw and PS, convert them to bhp for consistency

In [29]:

```
for index, rows in data.iterrows():
    if "bhp" in
str(data.iloc[index, data.columns.get_loc("power")]).lower():
    data.iloc[index, data.columns.get_loc("power")] =
data.iloc[index, data.columns.get_loc("power")].replace("bhp", ''
)
    if "hp" in
str(data.iloc[index, data.columns.get_loc("power")]).lower():
    data.iloc[index, data.columns.get_loc("power")] =
float(data.iloc[index, data.columns.get_loc("power")].split("
")[0].replace("hp", "").split("-")[0]) * 0.986
    if "kw" in
str(data.iloc[index, data.columns.get_loc("power")]).lower():
    data.iloc[index, data.columns.get_loc("power")] =
float(data.iloc[index, data.columns.get_loc("power")].split("
")[0].lower().replace("kw", "")) * 1.341
    if "ps" in
str(data.iloc[index, data.columns.get_loc("power")]).lower():
    data.iloc[index, data.columns.get_loc("power")] =
```

```
float(data.iloc[index,data.columns.get_loc("power")].split("
")[0].lower().replace("ps","")) * 0.99

    if "@" in
str(data.iloc[index,data.columns.get_loc("power")].lower():
        data.iloc[index,data.columns.get_loc("power")] =
data.iloc[index,data.columns.get_loc("power")].split("@")[0].st
rip()
```

In [30]:

```
data.isna().sum()
```

Out[30]:

model_name	0
model_year	0
kms_driven	0
owner	0
location	0
mileage	0
power	0
price	0
cc	0
brand	0

```
dtype: int64
```

Preprocessing KMS driven

```
In [31]:
```

```
data
```

```
Out[31]:
```

	model_name	model_year	kms_driven	owner	location	mileage	power	price	cc	brand
0	Bajaj Avenger Cruise 220 2017	2017	17000 Km	first owner	hyderabad	35	19	63500	220	Bajaj
1	Royal Enfield Classic 350cc 2016	2016	50000 Km	first owner	hyderabad	35	19.80	115000	350	Royal
2	Hyosung GT250R 2012	2012	14795 Km	first own	hyderabad	30	28	300000	25	Hyosung

				er					0	
3	Bajaj Dominar 400 ABS 2017	2017	Mileage 28 Kms	first own er	pondic herry	28	34. 50	100 000	4 0 0	Baja j
4	Jawa Perak 330cc 2020	2020	2000 Km	first own er	banga lore	37.59 9995	30	197 500	3 3 0	Jaw a
...
78 52	Yamaha YZF-R15 150cc 2011	2011	7000 Km	first own er	agra	42	16	550 00	1 5 0	Yam aha
78 53	Bajaj Discover 100cc 2015	2015	Mileage 80 Kmpl	first own er	delhi	80	7.7	280 00	1 0 0	Baja j
78	Bajaj Pulsar	2016	6407	first own	banga	65	17	617	1 8	Baja

54	180cc 2016		Km	er	lore			40	0	j
78 55	Bajaj V15 150cc 2016	2016	7524 Km	first own er	banga lore	57	11. 80	490 00	1 5 0	Baja j
78 56	Bajaj Pulsar 220cc 2016	2016	15000 Km	first own er	chenn ai	38	21	650 00	2 2 0	Baja j

7857 rows × 10 columns

Distance driven is represented as a string with units. Remove the units and keep only the distance value.

```
In [32]:
kms = data.kms_driven.to_numpy()
for i in range(len(kms)):
    kms[i] = kms[i].split(" ")[0]
```

There are inconsistent entries within the kms_drive column - few are given as mileage value and others are just "yes" strings. Replace them with average distance driven for a standard used bike.

```
In [33]:
```

```
for i in range(len(kms)):
    if str(kms[i]).lower() in ["mileage", "yes"]:
        kms[i] = np.nan
data['kms_driven'] = kms
mean =
kms[~np.isnan(kms.astype('float'))].astype('float').mean()
data.kms_driven.fillna(value = mean, inplace=True)
```

Fixing Data Types

Depending on column, cast each one to respective data type like float, int or string.

In [34]:

```
data.isna().sum()
```

Out[34]:

model_name	0
model_year	0
kms_driven	0
owner	0
location	0
mileage	0
power	0
price	0

```
cc          0
brand       0
```

```
dtype: int64
```

```
In [35]:
```

```
data = data.astype({"model_name" : str,
                    "model_year" : int,
                    "kms_driven" : int,
                    "owner"      : str,
                    "location"   : str,
                    "mileage"    : float,
                    "power"      : float,
                    "price"      : int,
                    "cc"         : int,
                    "brand"      : str})
```

```
In [36]:
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 7857 entries, 0 to 7856
```

```
Data columns (total 10 columns):
```

```
#      Column      Non-Null Count  Dtype
```

```
-----
0    model_name    7857 non-null    object
1    model_year    7857 non-null    int64
2    kms_driven    7857 non-null    int64
3    owner         7857 non-null    object
4    location      7857 non-null    object
5    mileage       7857 non-null    float64
6    power         7857 non-null    float64
7    price         7857 non-null    int64
8    cc           7857 non-null    int64
9    brand         7857 non-null    object
```

```
dtypes: float64(2), int64(4), object(4)
```

```
memory usage: 614.0+ KB
```

Exploratory Data Analysis

Statistics

```
In [37]:
```

```
pd.set_option("display.float",str)
```

```
data.describe().drop(["model_year"],axis=1).round(2)
```

```
Out[37]:
```


	kms_dr iven	mile age	pow er	price	cc
cou nt	7857.0	7857 .0	785 7.0	7857.0	785 7.0
me an	23090. 09	44.6 8	20.8 3	10679 1.34	254. 08
std	24813. 44	16.4 2	15.0 9	13892 6.12	182. 84
min	0.0	5.0	7.0	0.0	97.0
25 %	11204. 0	35.0	14.0	42000. 0	150. 0
50 %	23090. 0	40.0	19.0	75000. 0	200. 0

75 %	25000. 0	55.0	23.8 6	12500 0.0	350. 0
ma x	10000 00.0	104. 0	197. 3	30000 00.0	245 8.0

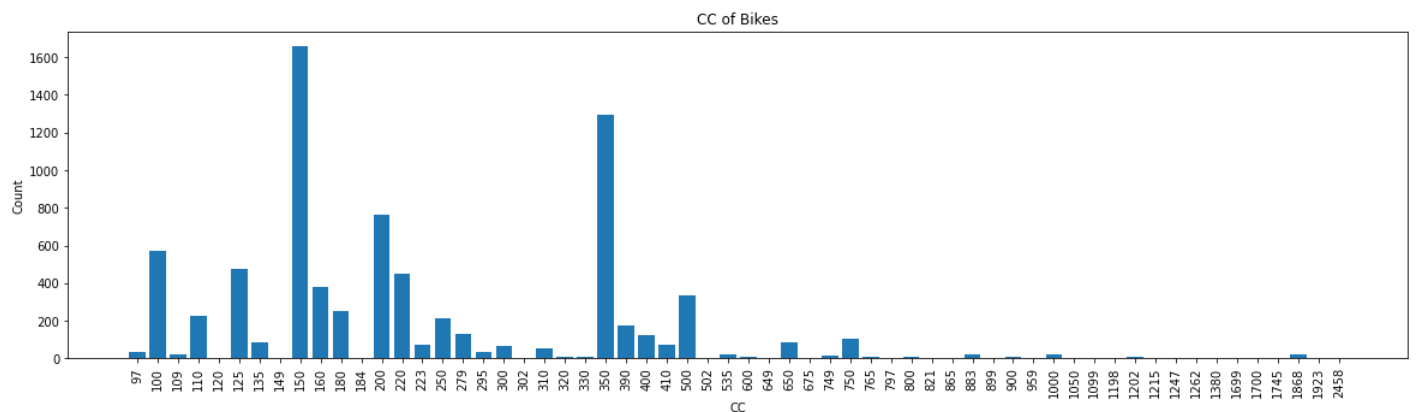
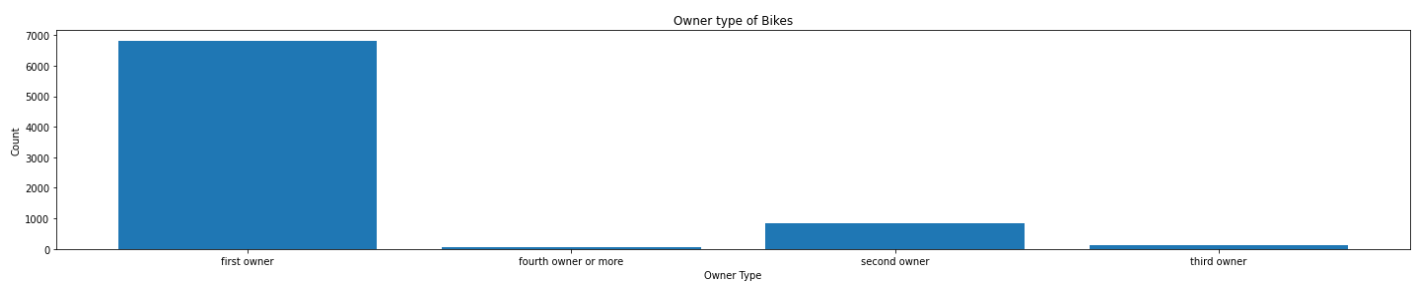
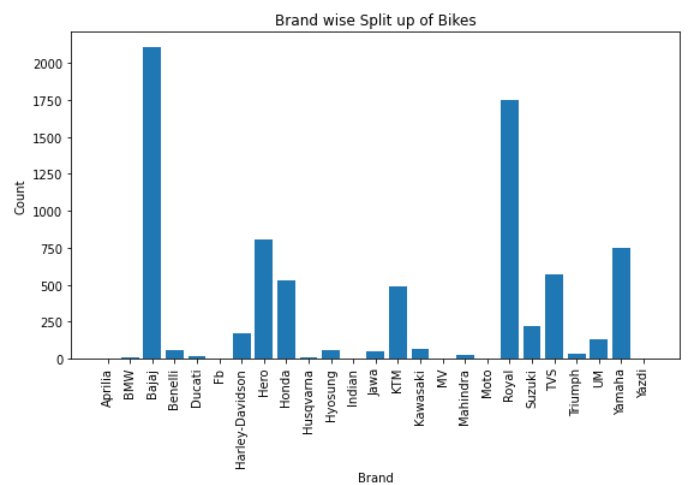
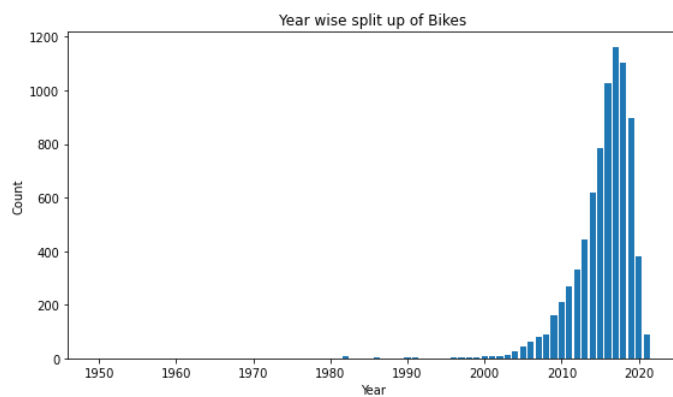
In [38]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
x,count = np.unique(data.model_year,return_counts=True)
plt.bar(x,count)
plt.title("Year wise split up of Bikes")
plt.xlabel("Year")
plt.ylabel("Count")

plt.subplot(1,2,2)
x,count = np.unique(data.brand,return_counts=True)
plt.bar(x,count)
plt.title("Brand wise Split up of Bikes")
plt.xlabel("Brand")
plt.xticks(rotation=90)
plt.ylabel("Count")
```

```
plt.figure(figsize=(20,4))
x,count = np.unique(data.owner,return_counts=True)
plt.bar(x,count)
plt.title("Owner type of Bikes")
plt.xlabel("Owner Type")
plt.ylabel("Count")
plt.tight_layout()
plt.show()

plt.figure(figsize=(20,5))
x,count = np.unique(data.cc,return_counts=True)
plt.bar([str(i) for i in x],count)
plt.title("CC of Bikes")
plt.xlabel("CC")
plt.xticks(rotation=90)
plt.ylabel("Count")
plt.show()
```



linkcode

Inference : The year wise split up shows a skewed graph with the number of models rising from the year 2000 with 2015 being the highest and the number getting floored from there onwards until 2020. Brands belonging to Bajaj and Royal Enfield are the highest in stock followed by Yamaha and Hero. The majority of bikes belong to First Owner which means it has been owned only by one customer. 150 cc and 350 cc

bikes are the highest in number followed by other common capacities like 200 cc and below. 500 cc stands out in the mid range power. Superbikes (1000 cc and above) are very less in count comparatively.

[Reference link](#)