Used Bike Prices

July 31, 2025

1 Used Bike Prices

1.1 Project Overview

This project aims to conduct an Exploratory Data Analysis (EDA) on a dataset of used bikes to understand the various factors influencing their prices in the Indian market. By cleaning, transforming, and visualizing the data, we will uncover key relationships between bike features (like mileage, power, and age) and their selling price. The insights gained will be valuable for potential buyers, sellers, and market analysts in the used bike industry.

1.2 Dataset Overview

- **model_name:** The specific model name of the bike, which may also contain details about its year and engine type.
- **model_year:** The manufacturing year of the bike, indicating its age.
- **kms_driven:** The total distance (in kilometers) the bike has traveled.
- **owner:** Categorical variable indicating the bike's ownership history (e.g., 'first owner', 'second owner').
- **location:** The geographical location of the bike's seller.
- **mileage:** The average fuel efficiency of the bike, expressed in kilometers per liter (kmpl).
- **power:** The engine's power output, primarily in Brake Horsepower (BHP).
- **price:** The target variable, representing the selling price of the used bike in Indian Rupees (INR).

1.3 Key Analytical Goals

- Calculate the age of the bike based on model_year and the current year.
- Analyze the distributions of kms_driven, mileage, power, price, and age
- Determine the most frequent brands, owner types, and locations to understand the common categories in the dataset.
- Investigate how kms_driven, model_year, mileage , and power individually influence the bike's price.
- Analyze how a combination of a bike's age and power together influences its price.

- Compare average price across different owner types (first, second, third owner) to assess if ownership history affects value and if bikes with fewer owners sell for more money.
- Analyze how price varies across different brands to see if certain brands are consistently more expensive or retain their value better.
- Identify the most frequently listed model_name and their average price, mileage, power, age to determine if certain models are typically more expensive or cheaper.
- Compare key metrics across the top 10 brands to highlight differences and competitive landscapes.
- Investigate if there are significant differences in price or other attributes based on location, specifically if bikes are priced differently in major cities versus smaller towns.
- Figure out what specific combination of features (e.g., low kilometers, high power, newer model year) makes a bike sell for a top price.
- Identify and understand any significant outliers in numerical columns, especially price and kms_driven, and investigate their characteristics

1.3.1 Load Required libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import plotly.express as px
import seaborn as sns
from datetime import date
import re
```

1.3.2 Load Data From CSV File

[2]:
```
# Load csv file
df = pd.read_csv("bikes.csv")
df.head(20)
```

[2]:
```
                                 model_name  model_year  \
0               Bajaj Avenger Cruise 220 2017        2017
1           Royal Enfield Classic 350cc 2016        2016
2                        Hyosung GT250R 2012        2012
3                  Bajaj Dominar 400 ABS 2017        2017
4                      Jawa Perak 330cc 2020        2020
5                      KTM Duke 200cc 2012        2012
6                   Bajaj Pulsar 180cc 2016        2016
7   TVS Apache RTR 200 4V Dual Channel ABS BS6 2020        2020
8                      KTM Duke 390cc 2018        2018
9                     Yamaha FZ16 150cc 2014        2014
10          Royal Enfield Classic 350cc 2018        2018
```

```
11             Royal Enfield Himalayan 410cc 2016        2016
12          Royal Enfield Bullet Electra 350cc 2017        2017
13                      Honda CB Shine 125cc 2018        2018
14            Royal Enfield Standard 350cc 2019        2019
15                        Bajaj Pulsar NS200 2018        2018
16             Royal Enfield Classic 350cc 2019        2019
17            Royal Enfield Standard 350cc 2015        2015
18              Royal Enfield Bullet 350cc 2016        2016
19        Royal Enfield Thunderbird 350cc ABS 2019        2019

        kms_driven        owner     location       mileage      power   price
0         17000 Km  first owner    hyderabad  \n\n 35 kmpl     19 bhp   63500
1         50000 Km  first owner    hyderabad  \n\n 35 kmpl  19.80 bhp  115000
2         14795 Km  first owner    hyderabad  \n\n 30 kmpl     28 bhp  300000
3     Mileage 28 Kms  first owner  pondicherry   \n\n 28 Kms  34.50 bhp  100000
4          2000 Km  first owner    bangalore        \n\n     30 bhp  197500
5         24561 Km  third owner    bangalore  \n\n 35 kmpl     25 bhp   63400
6         19718 Km  first owner    bangalore  \n\n 65 kmpl     17 bhp   55000
7   Mileage 40 Kmpl  first owner    hyderabad  \n\n 40 Kmpl  20.21 bhp  120000
8          1350 Km  first owner       jaipur  \n\n 25 kmpl  42.90 bhp  198000
9   Mileage 58 Kmpl  first owner    bangalore  \n\n 58 Kmpl     13 bhp   40000
10        25000 Km  first owner      chennai  \n\n 35 kmpl  19.80 bhp  136900
11        26240 Km  first owner    ghaziabad  \n\n 32 kmpl  24.50 bhp  112000
12        18866 Km  first owner        delhi  \n\n 40 kmpl   19.8 Bhp  110000
13  Mileage 65 Kmpl  first owner        delhi  \n\n 65 Kmpl     10 bhp   50000
14  Mileage 30 Kmpl  first owner        delhi  \n\n 30 Kmpl     18 bhp  131000
15  Mileage 42 Kmpl  third owner        delhi  \n\n 42 Kmpl  23.20 bhp   53000
16        12634 Km  first owner        delhi  \n\n 35 kmpl  19.80 bhp  160000
17  Mileage 37 Kmpl  first owner        delhi  \n\n 37 Kmpl  19.80 bhp  121000
18        13000 Km  first owner        delhi  \n\n 37 kmpl  19.80 bhp  111000
19        28000 Km  first owner        delhi  \n\n 40 kmpl  19.80 bhp  131500
```

1.3.3 Data Inspection

[3]:
```python
# Get all columns name
df.columns
```

[3]:
```
Index(['model_name', 'model_year', 'kms_driven', 'owner', 'location',
       'mileage', 'power', 'price'],
      dtype='object')
```

[4]:
```python
# get dataframe infomations like data type, counts and non-null
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7857 entries, 0 to 7856
Data columns (total 8 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   object
 3   owner       7857 non-null   object
 4   location    7838 non-null   object
 5   mileage     7846 non-null   object
 6   power       7826 non-null   object
 7   price       7857 non-null   int64
dtypes: int64(2), object(6)
memory usage: 491.2+ KB
```

Clean Data

[5]:
```python
# Remove all spaces from column names and make it lower case
df.columns = df.columns.str.strip().str.lower()

# convert data type of string.
def convert_dtype(string):

    # Ensure it's a string
    string = str(string)

    if '.' in string:
        return float(string)
    else:
        return int(string)

# Clean the 'kms_driven' column:
# 1. Remove 'Kmpl','kms' and 'km' as they are inconsistently present and not␣
 ↪needed for standardization.

df['kms_driven'] = df['kms_driven'].str.replace(r'(Kmpl|Kms|Km)', '',␣
 ↪regex=True, flags=re.IGNORECASE).str.strip().str.lower()

def clean_kms(kms_str):

    if type(kms_str) is str:
        # Check if string strat with mileage or not. if yes that make is NAN.
        if kms_str.startswith('mileage') or kms_str.startswith('yes'):
            return None
        else:
            return kms_str

df['kms_driven'] = df['kms_driven'].apply(clean_kms)

# Clean the 'mileage' column:
```

```
# 1. Remove 'Mileage','Kmpl','kms','km' and extra words as they are␣
↪inconsistently present and not needed for standardization.
# 2. From range values like 45-55 take max range 55 and remove 45-

df['mileage'] = df['mileage'].str.strip()

# remove non-numeric data from mileage
def clean_mileage(mileage_str):

    # remove all non-numeric data and keep only numeric(int and float) data
    mileage = ''.join(re.findall(r'[-+]?\d*\.?\d+', str(mileage_str)))

    # check if range value exiest
    if '-' in mileage:
        # split value by dash(-)
        parts = mileage.split('-')
        # Take the last part for max range
        return convert_dtype(parts[-1])
    else:
        if not mileage:
            return None
        else:
            return mileage

df['mileage'] = df['mileage'].apply(clean_mileage)

# Clean the 'power' column:
# 1. Convert 'kW' and 'PS' values to 'bhp' to standardize units.
# 2. Remove '@rpm' suffixes as they are inconsistently present and not needed␣
↪for power standardization.

kw_value = 1.34102
ps_value = 0.986

# Define Function to clear power columns and conver kw and ps to bhp
def clean_power(power_str):

    # Ensure it's a string and lowercase
    power_str = str(power_str).strip().lower()

    # Check for 'kw' and convert
    if 'kw' in power_str:
        value = float(re.findall(r'(\d+\.?\d*)', power_str)[0])
        return value * kw_value
    # Check for 'ps' and convert
    elif 'ps' in power_str:
        value = float(re.findall(r'(\d+\.?\d*)', power_str)[0])
```

```
        return value * ps_value
    else:
        # Extract numerical part for bhp and other cases
        match = re.findall(r'(\d+\.?\d*)', power_str)
        if match:
            return match[0]
        else:
            if not match:
                return None
            else:
                return match

df['power'] = df['power'].apply(clean_power)


# Drop Null values
df.dropna(inplace=True)

# Convert data type
df['kms_driven'] = df['kms_driven'].astype(int)
df['mileage'] = df['mileage'].astype(float).round(2)
df['power'] = df['power'].astype(float).round(2)

# Drop row if column have 0 value
df = df[df['kms_driven'] > 0].copy()
df = df[df['price'] > 0].copy()

df.head(20)
```

[5]:

```
                               model_name  model_year  kms_driven  \
0           Bajaj Avenger Cruise 220 2017        2017       17000
1          Royal Enfield Classic 350cc 2016        2016       50000
2                      Hyosung GT250R 2012        2012       14795
5                      KTM Duke 200cc 2012        2012       24561
6                  Bajaj Pulsar 180cc 2016        2016       19718
8                      KTM Duke 390cc 2018        2018        1350
10         Royal Enfield Classic 350cc 2018        2018       25000
11       Royal Enfield Himalayan 410cc 2016        2016       26240
12   Royal Enfield Bullet Electra 350cc 2017        2017       18866
16         Royal Enfield Classic 350cc 2019        2019       12634
18          Royal Enfield Bullet 350cc 2016        2016       13000
19  Royal Enfield Thunderbird 350cc ABS 2019        2019       28000
21         Royal Enfield Electra 350cc 2018        2018       23350
23           Bajaj Avenger Street 220 2016        2016        9551
24          Royal Enfield Bullet 350cc 2018        2018       23522
25      Royal Enfield Thunderbird 350cc 2015        2015       25000
26                 Yamaha SZ-RR 150cc 2012        2012       12000
```

```
27                   Yamaha FZs 150cc 2015        2015        10168
28            Royal Enfield Classic 350cc 2018        2018         7000
29            Royal Enfield Classic 350cc 2016        2016        12000

          owner   location  mileage  power   price
0   first owner  hyderabad     35.0   19.0   63500
1   first owner  hyderabad     35.0   19.8  115000
2   first owner  hyderabad     30.0   28.0  300000
5   third owner  bangalore     35.0   25.0   63400
6   first owner  bangalore     65.0   17.0   55000
8   first owner     jaipur     25.0   42.9  198000
10  first owner    chennai     35.0   19.8  136900
11  first owner  ghaziabad     32.0   24.5  112000
12  first owner      delhi     40.0   19.8  110000
16  first owner      delhi     35.0   19.8  160000
18  first owner      delhi     37.0   19.8  111000
19  first owner      delhi     40.0   19.8  131500
21  first owner      delhi     37.0   19.0  115000
23  first owner      delhi     53.0   19.0   52000
24  first owner   ludhiana     37.0   19.8  190000
25  first owner      delhi     40.0   19.8   67000
26  first owner      delhi     55.0   12.0   28000
27  first owner      delhi     45.0   13.0   44000
28  first owner      delhi     35.0   19.8  141500
29  first owner      delhi     35.0   19.8  110000
```

[6]:
```python
# get dataframe infomations like data type, counts and non-null
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5054 entries, 0 to 7856
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   model_name  5054 non-null   object
 1   model_year  5054 non-null   int64
 2   kms_driven  5054 non-null   int64
 3   owner       5054 non-null   object
 4   location    5054 non-null   object
 5   mileage     5054 non-null   float64
 6   power       5054 non-null   float64
 7   price       5054 non-null   int64
dtypes: float64(2), int64(3), object(3)
memory usage: 355.4+ KB
```

[7]:
```python
# Check null value in dataframe
df.isnull().sum()
```

[7]:
```
model_name    0
model_year    0
kms_driven    0
owner         0
location      0
mileage       0
power         0
price         0
dtype: int64
```

[8]:
```
# Check duplicate value in dataframe
df.duplicated().sum()
```

[8]:
```
np.int64(0)
```

[9]:
```
# get descriptive statistics of a dataframe like the central tendency,␣
↪dispersion, and shape of a distribution
df.describe().T
```

[9]:
```
             count           mean            std     min       25%      50%  \
model_year  5054.0    2015.203403       3.689484  1970.0   2014.00   2016.0
kms_driven  5054.0   24048.467155   29508.517485     3.0   9769.25  18000.0
mileage     5054.0      41.826296      15.986436     5.0     35.00     38.0
power       5054.0      21.981609      16.760427     7.0     14.00     19.8
price       5054.0  115655.828651  143373.851083  2000.0  45000.00  80400.0

                 75%        max
model_year    2018.0     2021.0
kms_driven   30500.0  1000000.0
mileage         53.0       95.0
power           24.6      197.3
price       133500.0  1900000.0
```

1.3.4 Feature Engineering

[10]:
```
# Calculate Age of Bike based on model_year and the current year.
df['age'] = date.today().year - df['model_year']

def clean_model_name(model_name):

    # Convert to lowercase for consistent processing
    cleaned_name = model_name.lower()

    # 1. Remove year (e.g., " 2017", " 2020") - looks for 4 digits at the end␣
↪of the string
    # or 4 digits preceded by a space
```

```
    cleaned_name = re.sub(r'\s*\d{4}$', '', cleaned_name) # Remove year at the␣
↪end
    cleaned_name = re.sub(r'\s*\d{4}\s*', ' ', cleaned_name) # Remove year in␣
↪the middle

    # Remove extra spaces and strip leading/trailing spaces
    cleaned_name = re.sub(r'\s+', ' ', cleaned_name).strip()

    return cleaned_name

# Apply the cleaning function to the 'model_name' column
df['model'] = df['model_name'].apply(clean_model_name)


# Common bikes brand names.
india_bikes_brand = [
    "Royal Enfield",
    "Jawa",
    "Yezdi",
    "Rajdoot",
    "Bajaj",
    "API",
    "Kinetic",
    "TVS",
    "Hero",
    "Honda",
    "Yamaha",
    "Suzuki",
    "KTM",
    "Harley-Davidson",
    "Hyosung",
    "Triumph",
    "BMW",
    "Kawasaki",
    "Aprilia",
    "Benelli",
    "Ducati",
    "Indian Motorcycle",
    "Mahindra",
    "Ather",
    "Revolt",
    "Tork",
    "Ultraviolette",
    "Komaki",
    "Avan",
    "Ampere",
    "Crayon",
```

```python
    "Odysse",
    "Yulu",
    "Husqvarna",
    "CFMoto",
    "Moto Morini",
    "Moto Guzzi",
    "MV Agusta",
    "Keeway",
    "LML",
    "UM"
]

india_bikes_brand = [brand.lower() for brand in india_bikes_brand]

def extract_brand(model_name):

    # Ensure it's a string and lowercase
    model_name = str(model_name).lower()

    for brand in india_bikes_brand:

        if brand in model_name:
            # Return the brand name
            return brand.title()

    # If no brand is found
    return "Other/Unknown"

# Apply the function to your DataFrame column
df['brand'] = df['model_name'].apply(extract_brand)

df.head(20)
# df.to_csv('bikes_cleaned.csv', index=False)
```

[10]:

```
                                 model_name  model_year  kms_driven  \
0              Bajaj Avenger Cruise 220 2017        2017       17000
1           Royal Enfield Classic 350cc 2016        2016       50000
2                        Hyosung GT250R 2012        2012       14795
5                        KTM Duke 200cc 2012        2012       24561
6                     Bajaj Pulsar 180cc 2016        2016       19718
8                        KTM Duke 390cc 2018        2018        1350
10          Royal Enfield Classic 350cc 2018        2018       25000
11        Royal Enfield Himalayan 410cc 2016        2016       26240
12   Royal Enfield Bullet Electra 350cc 2017        2017       18866
16          Royal Enfield Classic 350cc 2019        2019       12634
18           Royal Enfield Bullet 350cc 2016        2016       13000
19  Royal Enfield Thunderbird 350cc ABS 2019        2019       28000
```

```
21          Royal Enfield Electra 350cc 2018        2018       23350
23             Bajaj Avenger Street 220 2016        2016        9551
24           Royal Enfield Bullet 350cc 2018        2018       23522
25     Royal Enfield Thunderbird 350cc 2015        2015       25000
26                  Yamaha SZ-RR 150cc 2012        2012       12000
27                    Yamaha FZs 150cc 2015        2015       10168
28          Royal Enfield Classic 350cc 2018        2018        7000
29          Royal Enfield Classic 350cc 2016        2016       12000

          owner   location  mileage  power   price  age  \
0   first owner  hyderabad     35.0   19.0   63500    8
1   first owner  hyderabad     35.0   19.8  115000    9
2   first owner  hyderabad     30.0   28.0  300000   13
5   third owner  bangalore     35.0   25.0   63400   13
6   first owner  bangalore     65.0   17.0   55000    9
8   first owner     jaipur     25.0   42.9  198000    7
10  first owner    chennai     35.0   19.8  136900    7
11  first owner  ghaziabad     32.0   24.5  112000    9
12  first owner      delhi     40.0   19.8  110000    8
16  first owner      delhi     35.0   19.8  160000    6
18  first owner      delhi     37.0   19.8  111000    9
19  first owner      delhi     40.0   19.8  131500    6
21  first owner      delhi     37.0   19.0  115000    7
23  first owner      delhi     53.0   19.0   52000    9
24  first owner   ludhiana     37.0   19.8  190000    7
25  first owner      delhi     40.0   19.8   67000   10
26  first owner      delhi     55.0   12.0   28000   13
27  first owner      delhi     45.0   13.0   44000   10
28  first owner      delhi     35.0   19.8  141500    7
29  first owner      delhi     35.0   19.8  110000    9

                               model          brand
0              bajaj avenger cruise 220          Bajaj
1          royal enfield classic 350cc  Royal Enfield
2                     hyosung gt250r        Hyosung
5                     ktm duke 200cc            Ktm
6                  bajaj pulsar 180cc          Bajaj
8                     ktm duke 390cc            Ktm
10         royal enfield classic 350cc  Royal Enfield
11       royal enfield himalayan 410cc  Royal Enfield
12  royal enfield bullet electra 350cc  Royal Enfield
16         royal enfield classic 350cc  Royal Enfield
18          royal enfield bullet 350cc  Royal Enfield
19 royal enfield thunderbird 350cc abs  Royal Enfield
21         royal enfield electra 350cc  Royal Enfield
23            bajaj avenger street 220          Bajaj
24          royal enfield bullet 350cc  Royal Enfield
```

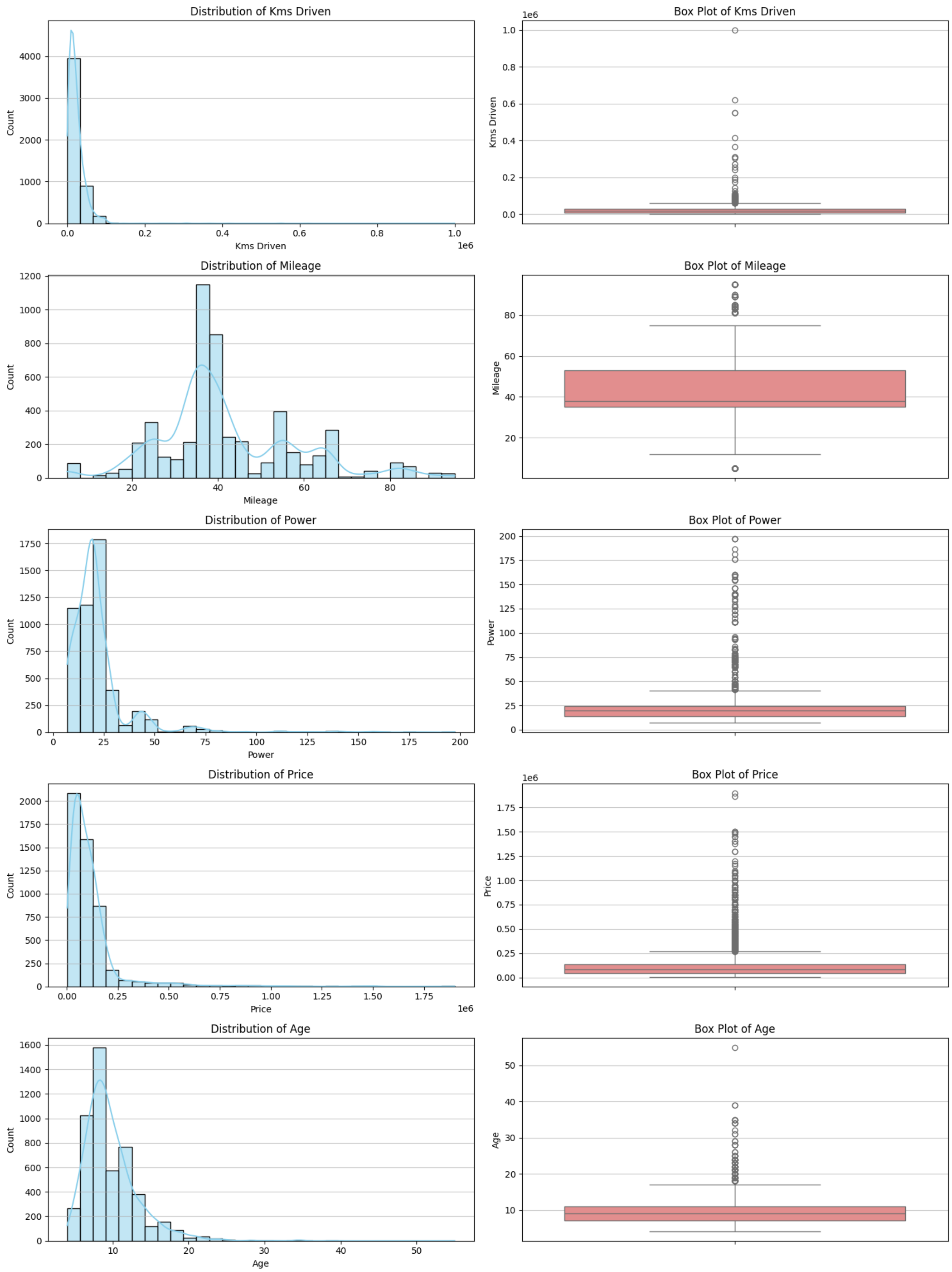
```
25      royal enfield thunderbird 350cc  Royal Enfield
26                   yamaha sz-rr 150cc         Yamaha
27                     yamaha fzs 150cc         Yamaha
28          royal enfield classic 350cc  Royal Enfield
29          royal enfield classic 350cc  Royal Enfield
```

1.4 Exploratory Analysis

1.4.1 Distributions Of Key Metrics

[101]:

```python
# List of numerical columns to analyze
numerical_col = ['kms_driven', 'mileage', 'power', 'price', 'age']

# set chart style properties
plt.figure(figsize=(15, 20))

# Loop through each numerical columns to create histograms and box plots
for i, feature in enumerate(numerical_col):

    # Create a subplot for the histogram
    plt.subplot(len(numerical_col), 2, 2*i + 1)

    sns.histplot(df[feature],
                 kde=True,
                 bins=30,
                 color='skyblue')

    plt.title(f'Distribution of {feature.replace("_", " ").title()}')
    plt.xlabel(feature.replace("_", " ").title())
    plt.grid(axis='y', alpha=0.75)

    # Create a subplot for the box plot
    plt.subplot(len(numerical_col), 2, 2*i + 2)

    sns.boxplot(y=df[feature]
                ,color='lightcoral')

    plt.title(f'Box Plot of {feature.replace("_", " ").title()}')
    plt.ylabel(feature.replace("_", " ").title())
    plt.grid(axis='y', alpha=0.75)

plt.tight_layout()
plt.show()
```



These charts are used to visualize the shape of the distribution of each key matrics.

- **Price:** is Likely a right-skewed distribution, with a high frequency of bikes in the lower to midprice ranges. The tail extends significantly to the right, indicating fewer but very expensive bikes. The box plot would confirm this skewness and highlight numerous high-value outliers, representing premium, luxury, or high-performance models.

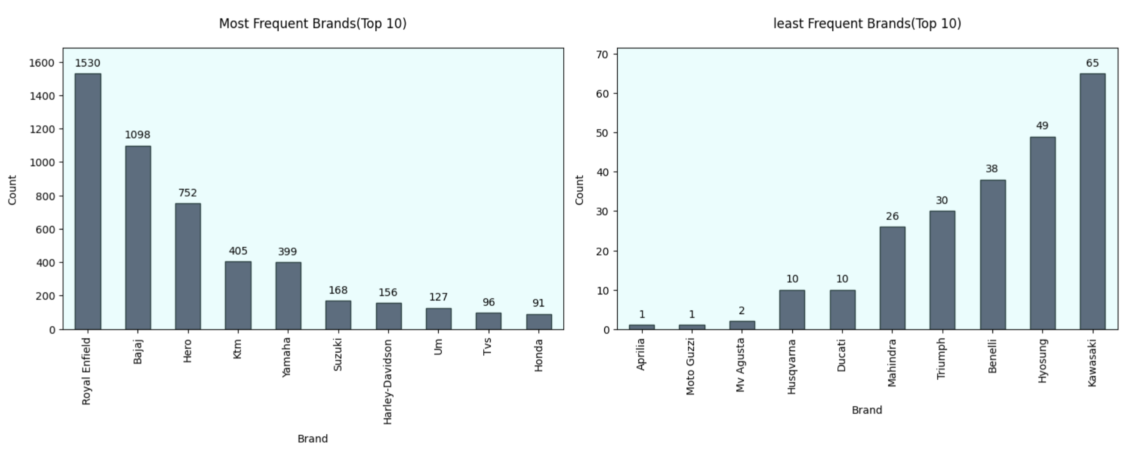
- **Kms Driven:** is Probably a right-skewed distribution, with most bikes having covered relatively low to moderate distances. The box plot would show a concentration of data at lower kilometers and a significant number of outliers at very high kilometer readings, suggesting bikes used for commercial purposes or very old, well-maintained vehicles.
- **Mileage:** is a multi-modal or slightly skewed distribution, with peaks around common mileage figures for commuter bikes. The box plot would show the central tendency and spread, with potential outliers for extremely fuel-efficient small bikes or very low-mileage performance bikes.
- **Power:** is expected to be right-skewed, with a large number of bikes having lower power, characteristic of the dominant commuter segment. The box plot would reveal outliers at higher power levels, corresponding to sports bikes, cruisers, and other high-performance categories.
- **Age:** is show a left-skewed distribution, with a higher concentration of newer bikes. The box plot would confirm this, with fewer very old bikes appearing as outliers.

The market is dominated by affordable, moderately used, and lower-powered bikes. High-value and high-usage bikes represent smaller, distinct segments often appearing as outliers.

1.4.2 Most And least Frequent Brands

[100]:
```
brand_count = df['brand'].value_counts()

# set chart style properties
plt.figure(figsize=(15, 6))

# Create a subplot
plt.subplot(1,2,1)

ax = sns.barplot(data=brand_count.head(10).reset_index(),
            x="brand",
            y="count",
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set barplot properties
ax.margins(y=0.10)
ax.set_facecolor("#ebfdfd")
ax.set_title('Most Frequent Brands(Top 10)',y=1.05)
ax.set_xlabel('Brand',labelpad=10)
ax.tick_params(axis='x', labelbottom=True,rotation=90)
ax.set_ylabel('Count',labelpad=10)
```

```python
# set count value on lagend
for container in ax.containers:
    ax.bar_label(container,
                 padding=5)

# Create a subplot
plt.subplot(1,2,2)

ax1 = sns.barplot(data=brand_count.tail(10).reset_index().
↪sort_values(by='count', ascending=True),
            x="brand",
            y="count",
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set barplot properties
ax1.margins(y=0.10)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('least Frequent Brands(Top 10)',y=1.05)
ax1.set_xlabel('Brand',labelpad=10)
ax1.tick_params(axis='x', labelbottom=True,rotation=90)
ax1.set_ylabel('Count',labelpad=10)

# set count value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



Most Frequent Brands

This chart shows the top 10 most common bike brands.

- **Royal Enfield** is by far the most frequent brand, with 1530 listings.
- **Bajaj** is the second most frequent, with 1098 listings.
- **Hero** follows with 752 listings.
- **Ktm** and **Yamaha** are next, with 405 and 399 listings respectively.
- The remaining brands in the top 10 (Suzuki, Harley-Davidson, Um, TVs, Honda) have significantly fewer listings, ranging from 168 down to 91.

Least Frequent Brands

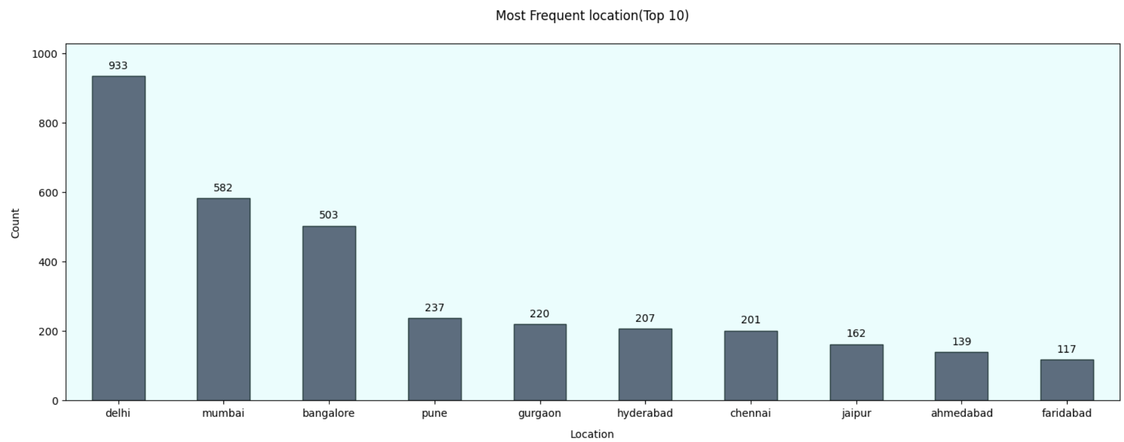
This chart shows the 10 least common bike brands.

- Brands like **Aprilia, Moto Guzzi, and Mv Agusta** have extremely low counts, indicating their rarity in the used market.
- Brands like **Husqvarna, Ducati, and Mahindra** have slightly higher but still very low counts.
- **Triumph, Benelli, Hyosung, and Kawasaki** are at the higher end of this "least frequent" group, with counts ranging from 30 to 65.

1.4.3 Most Frequent Locations

[99]:
```python
location_count = df['location'].value_counts()

# set chart style properties
plt.figure(figsize=(15, 6))

ax = sns.barplot(data=location_count.head(10).reset_index(),
            x="location",
            y="count",
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set barplot properties
ax.margins(y=0.10)
ax.set_facecolor("#ebfdfd")
ax.set_title('Most Frequent location(Top 10)',y=1.05)
ax.set_xlabel('Location',labelpad=10)
ax.set_ylabel('Count',labelpad=10)
```

```python
# set count value on lagend
for container in ax.containers:
    ax.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



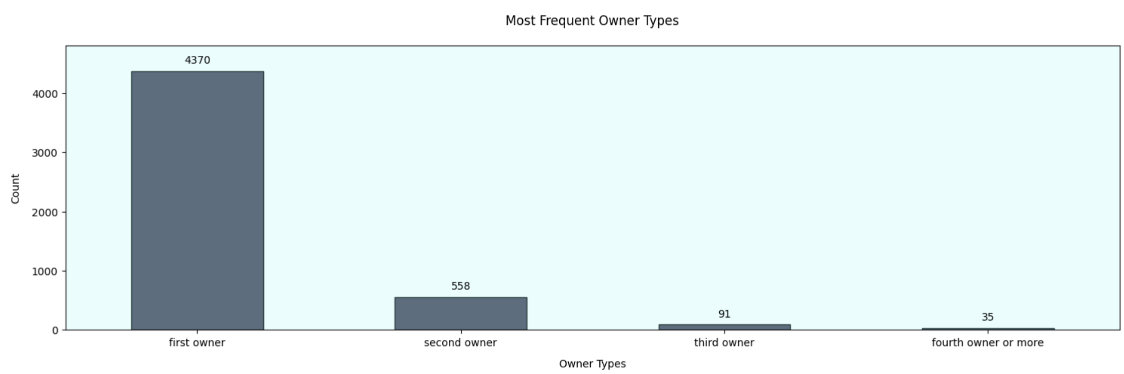
This chart presents the top 10 cities with the highest number of used bike listings.

- **Delhi** has the highest number of listings, with 933 bikes.
- **Mumbai** is second, with 582 listings.
- **Bangalore** follows closely with 503 listings.
- **Pune and Gurgaon** are next, with 237 and 220 listings respectively.
- The remaining cities in the top 10 (Hyderabad, Chennai, Jaipur, Ahmedabad, Faridabad) have counts ranging from 207 down to 117.

1.4.4 Most Frequent Owner Types

[98]:
```python
owner_count = df['owner'].value_counts()

# set chart style properties
plt.figure(figsize=(15, 5))

ax = sns.barplot(data=owner_count.head(10).reset_index(),
            x="owner",
            y="count",
            color="#586d83",
            errorbar = None,
```

```python
            width = 0.5,
            edgecolor="#2b4141"
           )

# set barplot properties
ax.margins(y=0.10)
ax.set_facecolor("#ebfdfd")
ax.set_title('Most Frequent Owner Types',y=1.05)
ax.set_xlabel('Owner Types',labelpad=10)
ax.set_ylabel('Count',labelpad=10)

# set count value on lagend
for container in ax.containers:
    ax.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



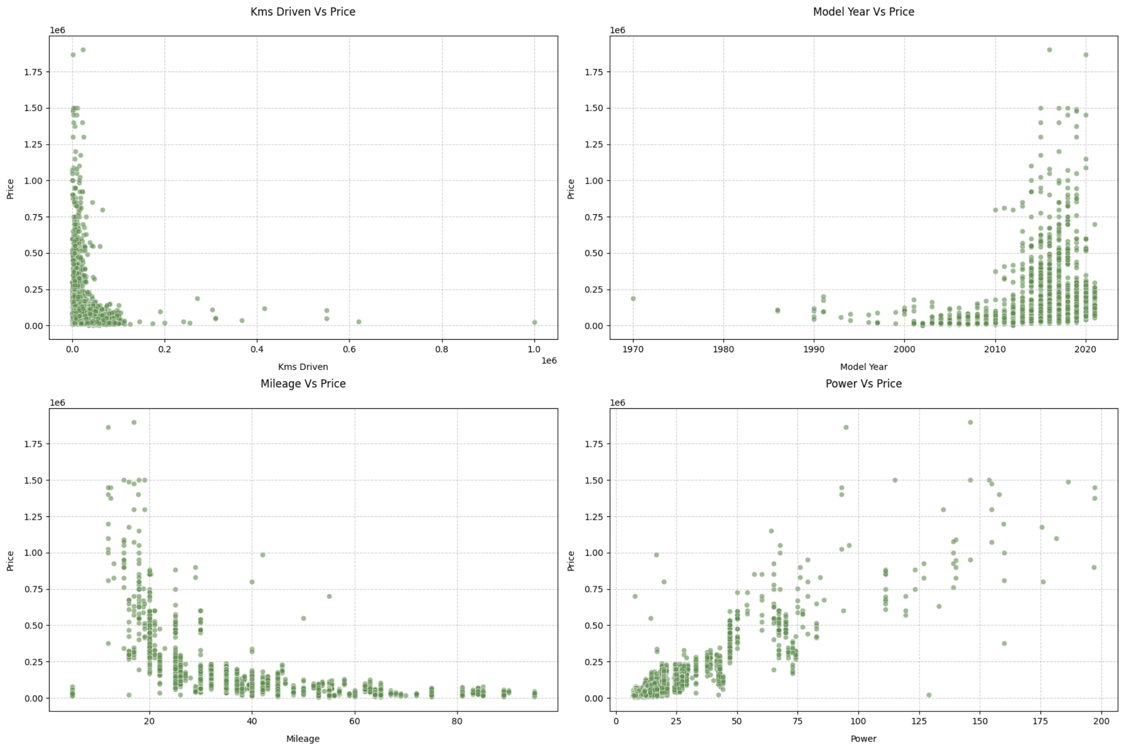
This chart shows the frequency of different owner types.

- **first owner** bikes are overwhelmingly the most common, with 4370 listings.
- **second owner** bikes are significantly fewer, with 558 listings.
- **third owner** bikes drop further to 91 listings.
- **fourth owner or more** bikes are the least common, with only 35 listings.

This chart clearly indicates that the vast majority of used bikes listed are from their first owner. This strong preference for **first owner** bikes highlights a key buyer sentiment in the used bike market, likely driven by perceived better condition, maintenance, and lower risk compared to bikes with multiple previous owners. Bikes with three or more owners are extremely rare.

1.4.5 Key Metrics Vs Price

[15]:
```python
# key metrics columns
key_metrics = ['kms_driven', 'model_year', 'mileage', 'power']

# set chart style properties
plt.figure(figsize=(18, 12))

# Loop through each key metrics to create a scatter plot against 'price'
for i, key in enumerate(key_metrics):

    # Create Subplot
    plt.subplot(2, 2, i + 1)

    # Create scatterplot
    sns.scatterplot(x=df[key],
                    y=df['price'],
                    alpha=0.6,
                    color='#5F8D4E')

    # set scatterplot properties
    plt.title(f'{key.replace("_", " ").title()} Vs Price',y=1.05)
    plt.xlabel(key.replace("_", " ").title(),labelpad=10)
    plt.ylabel('Price',labelpad=10)
    plt.grid(True, linestyle='--', alpha=0.6)

plt.tight_layout()
plt.show()
```



Kms Driven Vs Price

The plot shows a general inverse relationship: as the kilometers driven increase, the price of the bike tends to decrease. There's a dense cluster of points at lower Kms Driven values with a wide range of prices, and as Kms Driven goes up, the price range narrows and generally drops. There are some outliers with very high Kms Driven but still relatively low prices.

Model Year Vs Price

This plot shows a clear positive correlation: as the Model Year increases, its Price generally increases. There's a noticeable upward trend, with the highest prices concentrated among bikes from more recent model years. Older bikes are typically found at much lower price points.

Mileage Vs Price

This plot suggests a complex or weak inverse relationship. There's a dense cluster of bikes with higher Mileage values that are generally priced lower. Conversely, many of the higher-priced bikes appear to have lower Mileage. This indicates that high fuel efficiency is often associated with more affordable, commuter-segment bikes, while premium/performance bikes tend to have lower mileage.

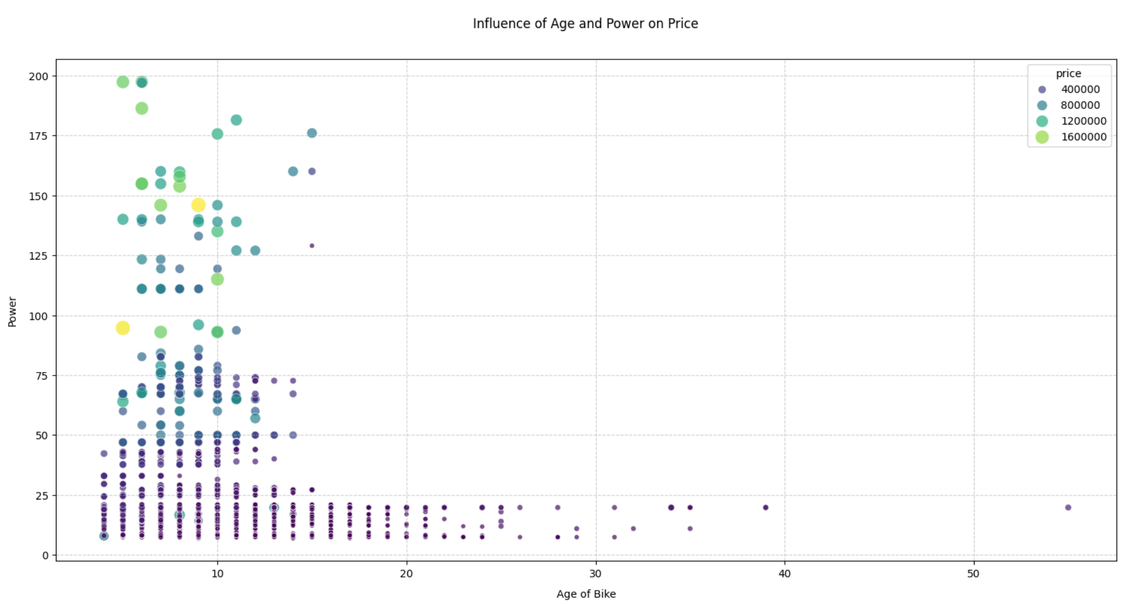
Power Vs Price

This plot shows a strong positive correlation: as the Power of the bike increases, its Price generally increases significantly. Bikes with higher Power values consistently command higher prices, forming an upward-sloping trend.

These scatter plots illustrate the individual influence of key features on bike prices. Model Year and Power show strong positive correlations with Price, indicating newer and more powerful bikes are more expensive. Conversely, Kms Driven generally has an inverse relationship with Price. Mileage shows an inverse trend, suggesting that higher-priced bikes often prioritize performance over fuel efficiency.

1.4.6 Influence of Age and Power on Price

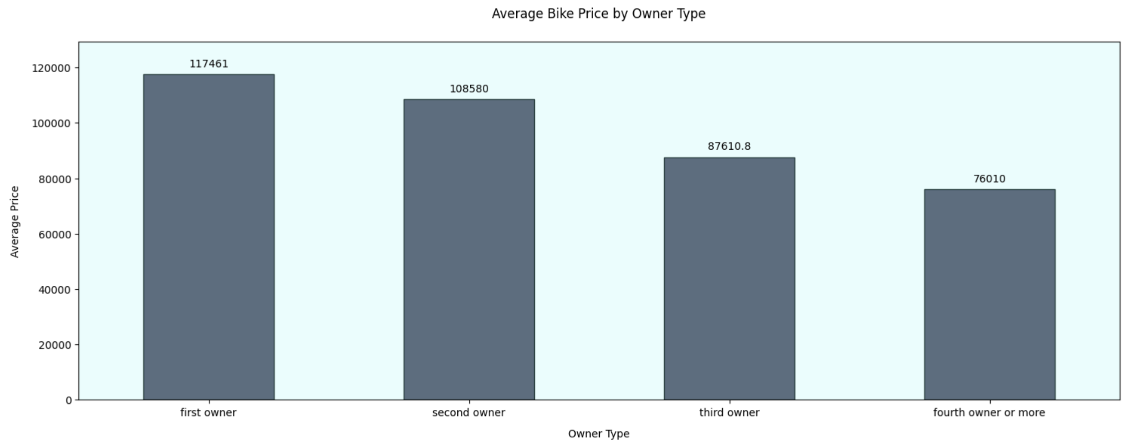
[95]:
```
# set chart style properties
plt.figure(figsize=(15, 8))

# Create scatterplot
scatter = sns.scatterplot(
    x=df['age'],
    y=df['power'],
    hue=df['price'],
    size=df['price'],
    sizes=(20, 200),
    palette='viridis',
    alpha=0.7,
    edgecolor='w',
    linewidth=0.5
)

plt.title('Influence of Age and Power on Price',y=1.05)
plt.xlabel('Age of Bike',labelpad=10)
plt.ylabel('Power',labelpad=10)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



This chart effectively visualizes how both Age and Power jointly influence a bike's Price. It clearly demonstrates that newer, high-power bikes command the highest prices. While price generally decreases with Age, higher Power helps bikes retain more value over time. Conversely, older and less powerful bikes are consistently found at the lowest price points.

1.4.7 Average Bike Price by Owner Type

[96]:
```
# Calculate the average price for each owner type
average_price = df.groupby('owner')['price'].mean().
↪sort_values(ascending=False).reset_index().round(2)

# Create a barplot
plt.figure(figsize=(15, 6))
ax = sns.barplot(data=average_price,
            x='owner',
            y='price',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

ax.margins(y=0.10)
ax.set_facecolor("#ebfdfd")
ax.set_title('Average Bike Price by Owner Type',y=1.05)
ax.set_xlabel('Owner Type',labelpad=10)
ax.set_ylabel('Average Price',labelpad=10)

# set average value on lagend
for container in ax.containers:
    ax.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



The chart shows a clear inverse relationship between the number of owners and the average bike price:

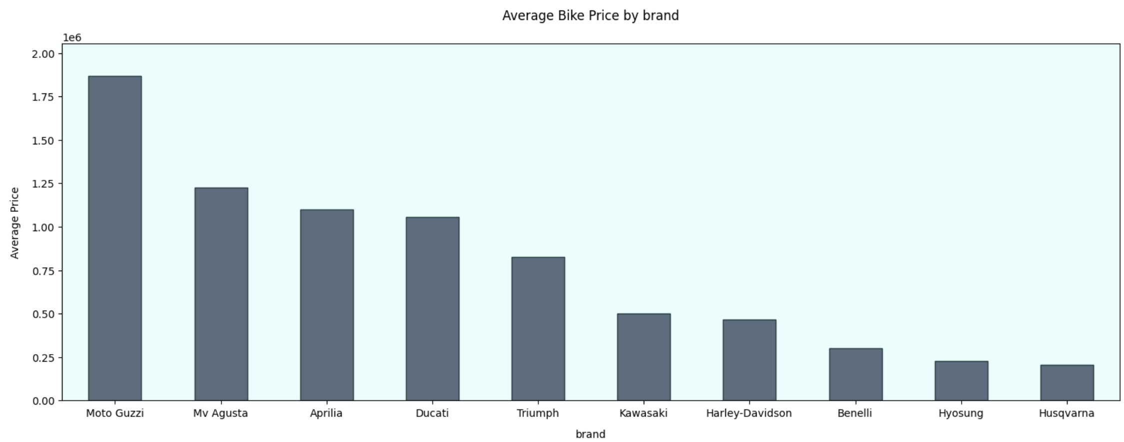
- **First owner** bikes have the highest average price at approximately 117,461.
- **Second owner** bikes are next, with an average price of around 108,580.
- **Third owner** bikes see a further drop to an average of about 87,610.8.
- **Fourth owner or more** bikes have the lowest average price, at approximately 76,010.

This chart clearly indicates that a bike's ownership history significantly impacts its resale value. Bikes with fewer previous owners consistently command higher average prices, with **first owner** bikes being the most valuable. This trend suggests that buyers perceive bikes with fewer owners as more reliable or better maintained, leading to a premium in the used market.

1.4.8 Average Bike Price by Brand (Top 10)

[97]:
```
# Calculate the average price for each owner type
average_price = df.groupby('brand')['price'].mean().
 ↪sort_values(ascending=False).reset_index().round(2)

# Create a barplot
plt.figure(figsize=(15, 6))

# Create barplot
ax = sns.barplot(data=average_price.head(10),
            x='brand',
            y='price',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )
```

```
ax.margins(y=0.10)
ax.set_facecolor("#ebfdfd")
ax.set_title('Average Bike Price by brand',y=1.05)
ax.set_xlabel('brand',labelpad=10)
ax.set_ylabel('Average Price',labelpad=10)

plt.tight_layout()
plt.show()
```



The chart shows the average price for several bike brands.

- **Moto Guzzi** has the highest average price, close to 1,900,000.
- **Mv Agusta and Aprilia** follow, with average prices around 1,200,000 and 1,100,000 respectively.
- **Ducati and Triumph** also show high average prices, above 800,000.
- Brands like **Kawasaki, Harley-Davidson, Benelli, Hyosung, and Husqvarna** have progressively lower average prices, ranging from approximately 500,000 down to around 200,000.

This chart clearly demonstrates a significant variation in average prices across different bike brands in the used market. Premium and niche brands like Moto Guzzi, Mv Agusta, Aprilia, and Ducati command substantially higher average prices, indicating their luxury positioning and strong value retention. In contrast, brands like Husqvarna and Hyosung, while still in the higher segment, have comparatively lower average prices within this group, highlighting distinct market segments and brand value perceptions.

1.4.9 Average Key Metrics Of Top 10 Models and Locations

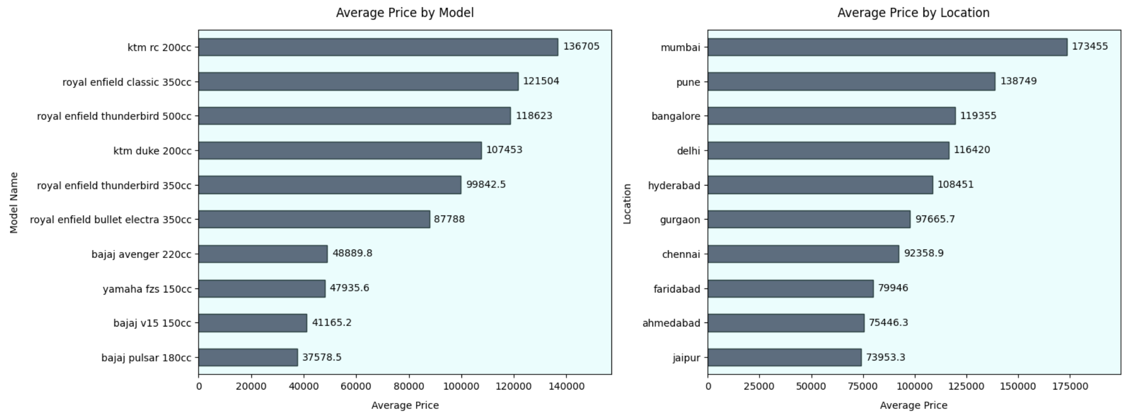
[77]:
```
# Calculate the frequent models count
frequent_models = df['model'].value_counts().head(10)
```

```
# creat frequent models datafream
frequent_models_df = df[df['model'].isin(frequent_models.index)].copy()

# Calculate the average price, mileage, power, and age for frequent models
model_average = frequent_models_df.groupby('model').agg(
    price=('price', 'mean'),
    kms=('kms_driven', 'mean'),
    mileage=('mileage', 'mean'),
    power=('power', 'mean'),
    age=('age', 'mean')
).sort_values(by='price', ascending=False).reset_index().round(2)

# Calculate the frequent models count
location_frequent_models = df['location'].value_counts().head(10)

# creat frequent models datafream
location_frequent_models_df = df[df['location'].isin(location_frequent_models.
 ↪index)].copy()

# Calculate the average price, mileage, power, and age for frequent models
location_model_average = location_frequent_models_df.groupby('location').agg(
    price=('price', 'mean'),
    kms=('kms_driven', 'mean'),
    mileage=('mileage', 'mean'),
    power=('power', 'mean'),
    age=('age', 'mean')
).sort_values(by='price', ascending=False).reset_index().round(2)
```

1.4.10 Top 10 Average Price by Model and location

[82]:
```
# set chart style properties
plt.figure(figsize=(16, 6))

# Set Subplot properties
plt.subplot(1,2,1)

# create price barplot
ax = sns.barplot(data=model_average,
            y='model',
            x='price',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set Price barplot properties
```

```python
ax.margins(x=0.15)
ax.set_facecolor("#ebfdfd")
ax.set_title('Average Price by Model',y=1.02)
ax.set_xlabel('Average Price',labelpad=10)
ax.set_ylabel('Model Name',labelpad=10)

# set price value on lagend
for container in ax.containers:
    ax.bar_label(container,
                 padding=5)

# Set Subplot properties
plt.subplot(1,2,2)

# create price barplot
ax1 = sns.barplot(data=location_model_average,
            y='location',
            x='price',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set Price barplot properties
ax1.margins(x=0.15)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average Price by Location',y=1.02)
ax1.set_xlabel('Average Price',labelpad=10)
ax1.set_ylabel('Location',labelpad=10)

# set price value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



Average Price by Model

This chart shows the average price for a top 10 bike models.

- **Ktm RC 200cc** has the highest average price among the displayed models, at 136,705.
- **Royal Enfield Classic 350cc and Royal Enfield Thunderbird 500cc** follow with average prices around 121,504 and 118,623 respectively.
- Other models like **Ktm Duke 200cc, Royal Enfield Thunderbird 350cc, and Royal Enfield Bullet Electra 350cc** are in the mid-range.
- Models such as **Bajaj Avenger 220cc, Yamaha FZS 150cc, Bajaj V15 150cc, and Bajaj Pulsar 180cc** have significantly lower average prices, ranging from approximately 48,889 down to 37,578.5.

This chart illustrates that specific bike models command significantly different average prices in the used market. High-performance models like KTM RC and popular Royal Enfield variants consistently fetch higher average prices. Conversely, mass-market commuter models from Bajaj and Yamaha are typically found at much lower average price points. This highlights the impact of model type and brand positioning on a bike's resale value.

Average Price by Location

This chart shows the average price of bikes across various locations(top 10).

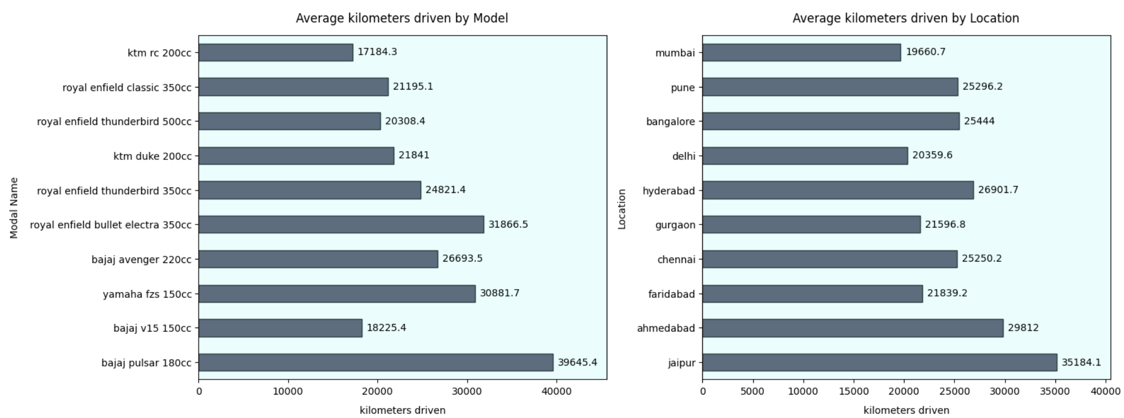
- **Mumbai** has the highest average bike price at 173,455.
- **Pune and Bangalore** follow with average prices around 138,749 and 119,355 respectively.
- **Delhi, Hyderabad, and Gurgaon** are in the mid-range of average prices.
- Cities like **Chennai, Faridabad, Ahmedabad, and Jaipur** show progressively lower average prices, with Jaipur having the lowest among the displayed locations at 73,953.3.

This chart demonstrates a clear geographical variation in average used bike prices. Major metropolitan cities such as Mumbai, Pune, and Bangalore exhibit the highest average selling prices for bikes. In contrast, cities like Jaipur and Ahmedabad show lower average prices. This suggests that local market demand, economic conditions, and possibly the availability of premium bikes vary significantly across different Indian cities.

1.4.11 Top 10 Average kilometers driven by Model and location

[135]:
```python
# set chart style properties
plt.figure(figsize=(16, 6))

# Set Subplot properties
plt.subplot(1,2,1)

# create kms barplot
ax = sns.barplot(data=model_average,
            y='model',
            x='kms',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set kms barplot properties
ax.margins(x=0.15)
ax.set_facecolor("#ebfdfd")
ax.set_title('Average kilometers driven by Model',y=1.02)
ax.set_xlabel('kilometers driven',labelpad=10)
ax.set_ylabel('Modal Name',labelpad=10)

# set kms value on lagend
for container in ax.containers:
    ax.bar_label(container,
                 padding=5)

# Set Subplot properties
plt.subplot(1,2,2)

# create kms barplot
ax1 = sns.barplot(data=location_model_average,
            y='location',
            x='kms',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set kms barplot properties
```

```
ax1.margins(x=0.15)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average kilometers driven by Location',y=1.02)
ax1.set_xlabel('kilometers driven',labelpad=10)
ax1.set_ylabel('Location',labelpad=10)

# set kms value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                  padding=5)

plt.tight_layout()
plt.show()
```



Average kilometers driven by Model

This chart shows the average kilometers driven for a top 10 bike models.

- **Ktm RC 200cc** has the lowest average KMS driven among the displayed models, at 17,164.3.
- **Royal Enfield Classic 350cc and Royal Enfield Thunderbird 500cc** follow with average KMS driven around 21,195.1 and 20,308.4 respectively.

Models like **Bajaj Pulsar 180cc** show a significantly higher average KMS driven at 39,645.4.

This chart illustrates that the average kilometers driven varies across different bike models. Performance-oriented models like KTM RC 200cc tend to have lower average usage, suggesting they might be ridden less frequently or for shorter distances. In contrast, some mass-market models like Bajaj Pulsar 180cc show significantly higher average kilometers, indicating more extensive use, possibly for daily commuting or longer rides.

Average kilometers driven by Location

This chart shows the average kilometers driven of bikes across various locations(top 10).

- **Mumbai** has the lowest average KMS driven at 19,660.7.

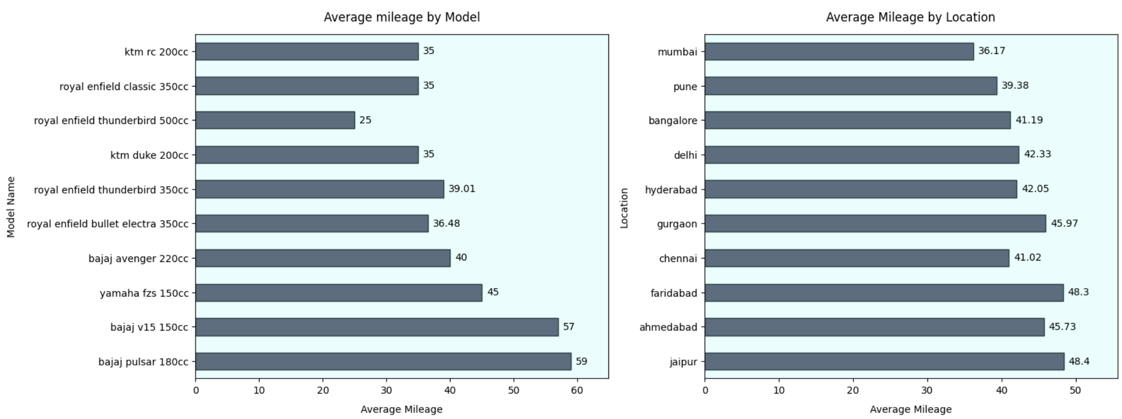
- **Delhi and Gurgaon** are also relatively low, around 20,359.6 and 21,596.8 respectively.
- **Jaipur and Ahmedabad** show significantly higher average KMS driven, at 35,184.1 and 29,812 respectively.

This chart highlights geographical differences in the average kilometers driven by bikes. Major metropolitan areas such as Mumbai and Delhi exhibit lower average kilometers driven, which could imply more frequent bike turnover or shorter daily commutes. Conversely, cities like Jaipur and Ahmedabad show higher average kilometers, suggesting bikes in these regions might undergo more extensive usage.

1.4.12 Top 10 Average Mileage by Model and location

[136]:
```
# set chart style properties
plt.figure(figsize=(16, 6))

# Set Subplot properties
plt.subplot(1,2,1)

# create mileage barplot
ax1 = sns.barplot(data=model_average,
            y='model',
            x='mileage',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set mileage barplot properties
ax1.margins(x=0.10)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average mileage by Model',y=1.02)
ax1.set_xlabel('Average Mileage',labelpad=10)
ax1.set_ylabel('Model Name',labelpad=10)

# set mileage value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

# Set Subplot properties
plt.subplot(1,2,2)

# create mileage barplot
ax1 = sns.barplot(data=location_model_average,
            y='location',
            x='mileage',
```

```
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set mileage barplot properties
ax1.margins(x=0.15)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average Mileage by Location',y=1.02)
ax1.set_xlabel('Average Mileage',labelpad=10)
ax1.set_ylabel('Location',labelpad=10)

# set mileage value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



Average mileage by Model

This chart shows the average mileage for a top 10 bike models.

- Models like **Bajaj Pulsar 180cc** and **Bajaj V15 150cc** have the highest average mileage among those displayed, at 59 kmpl and 57 kmpl respectively.
- **Yamaha FZS 150cc** and **Bajaj Avenger 220cc** are also relatively fuel-efficient, around 45 kmpl and 40 kmpl.
- Performance-oriented models like **Ktm RC 200cc, Royal Enfield Classic 350cc, and Ktm Duke 200cc** show lower average mileage, typically around 35 kmpl.
- **Royal Enfield Thunderbird 500cc** has the lowest average mileage among this group, at

25 kmpl.

This chart highlights that average mileage varies significantly across different bike models, generally reflecting their design purpose. Commuter-focused models like Bajaj Pulsar and V15 show higher fuel efficiency, while performance-oriented bikes such as KTMs and larger Royal Enfields have lower average mileage. This indicates a trade-off between power/performance and fuel economy, influencing buyer choices based on their priorities.

Average mileage by Location

This chart shows the average mileage of bikes across various locations(top 10).

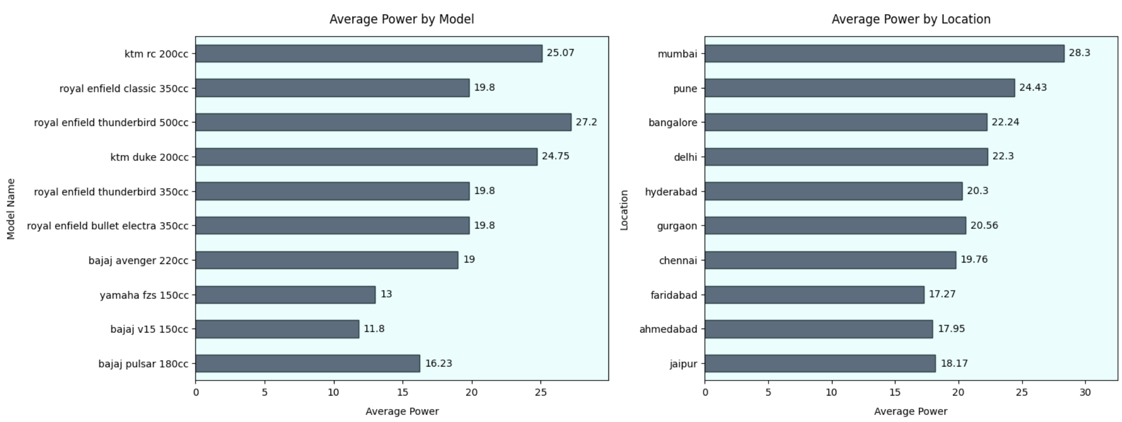
- **Mumbai** has the lowest average mileage among the displayed cities, at 36.17 kmpl.
- **Pune and Bangalore** also show relatively lower average mileages, around 39.38 kmpl and 41.19 kmpl.
- Cities like **Jaipur and Faridabad** exhibit higher average mileages, at 48.4 kmpl and 48.3 kmpl respectively.

This chart reveals geographical differences in the average mileage of bikes. Major metropolitan areas like Mumbai and Pune tend to have lower average mileages, possibly due to a higher prevalence of performance bikes or shorter city commutes where fuel efficiency is less critical. Conversely, cities like Jaipur and Faridabad show higher average mileages, suggesting a stronger market for more fuel-efficient commuter bikes in those regions.

1.4.13 Top 10 Average Power by Model and location

[137]:
```python
# set chart style properties
plt.figure(figsize=(16, 6))

# Set Subplot properties
plt.subplot(1,2,1)

# create power barplot
ax1 = sns.barplot(data=model_average,
            y='model',
            x='power',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set power barplot properties
ax1.margins(x=0.10)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average Power by Model',y=1.02)
ax1.set_xlabel('Average Power',labelpad=10)
ax1.set_ylabel('Model Name',labelpad=10)
```

```
# set power value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

# Set Subplot properties
plt.subplot(1,2,2)

# create power barplot
ax1 = sns.barplot(data=location_model_average,
            y='location',
            x='power',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set power barplot properties
ax1.margins(x=0.15)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average Power by Location',y=1.02)
ax1.set_xlabel('Average Power',labelpad=10)
ax1.set_ylabel('Location',labelpad=10)

# set power value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



Average Power by Model

This chart shows the average power for a top 10 bike models.

- **Ktm RC 200cc** and **Royal Enfield Thunderbird 500cc** have the highest average power among the displayed models, at 25.07 BHP and 27.2 BHP respectively.
- **Ktm Duke 200cc** also shows high power at 24.75 BHP.
- Models like **Royal Enfield Classic 350cc, Royal Enfield Thunderbird 350cc, Royal Enfield Bullet Electra 350cc, and Bajaj Avenger 220cc** are in a mid-range, around 19-20 BHP.
- **Yamaha FZS 150cc, Bajaj V15 150cc, and Bajaj Pulsar 180cc** have significantly lower average power, ranging from approximately 13 BHP down to 11.8 BHP.

This chart demonstrates that average engine power varies considerably across different bike models, directly reflecting their design and intended use. Performance-oriented models like KTMs and larger Royal Enfields exhibit significantly higher average power. In contrast, commuter-focused bikes such as Bajaj Pulsar and V15 models have much lower average power, indicating their emphasis on fuel efficiency and everyday usability rather than raw performance.

Average Power by Location

This chart shows the average power of bikes across various locations(top 10).

- **Mumbai** has the highest average power at 28.3 BHP.
- **Pune, Bangalore, and Delhi** also show relatively higher average powers, around 24.43 BHP, 22.24 BHP, and 22.3 BHP respectively.

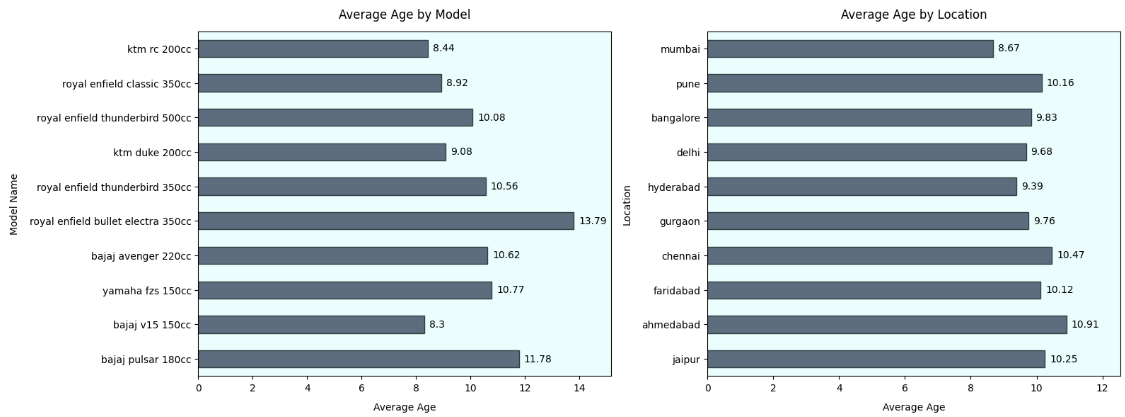
Cities like **Faridabad, Ahmedabad, and Jaipur** exhibit lower average powers, ranging from approximately 17.27 BHP down to 18.17 BHP.

This chart reveals geographical differences in the average engine power of bikes listed. Major metropolitan areas like Mumbai, Pune, and Bangalore tend to have higher average power, suggesting a greater prevalence of performance or higher-end bikes in these markets. Conversely, cities like Faridabad and Jaipur show lower average power, indicating a stronger presence of commuter or lower-powered bikes, possibly due to different market demands or economic factors..

1.4.14 Top 10 Average Age by Model and location

[89]:
```
# set chart style properties
plt.figure(figsize=(16, 6))

# Set Subplot properties
plt.subplot(1,2,1)

# create age barplot
ax1 = sns.barplot(data=model_average,
            y='model',
            x='age',
            color="#586d83",
```

```python
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set age barplot properties
ax1.margins(x=0.10)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average Age by Model',y=1.02)
ax1.set_xlabel('Average Age',labelpad=10)
ax1.tick_params(axis='x', labelbottom=True,rotation=0)
ax1.set_ylabel('Model Name',labelpad=10)

# set age value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

# Set Subplot properties
plt.subplot(1,2,2)

# create age barplot
ax1 = sns.barplot(data=location_model_average,
            y='location',
            x='age',
            color="#586d83",
            errorbar = None,
            width = 0.5,
            edgecolor="#2b4141"
           )

# set age barplot properties
ax1.margins(x=0.15)
ax1.set_facecolor("#ebfdfd")
ax1.set_title('Average Age by Location',y=1.02)
ax1.set_xlabel('Average Age',labelpad=10)
ax1.set_ylabel('Location',labelpad=10)

# set age value on lagend
for container in ax1.containers:
    ax1.bar_label(container,
                 padding=5)

plt.tight_layout()
plt.show()
```



Average age by Model

This chart shows the average price for a top 10 bike models.

- Models like **Bajaj V15 150cc and Ktm RC 200cc** have the lowest average age among those displayed, at 8.3 years and 8.44 years respectively.
- **Royal Enfield Classic 350cc and Ktm Duke 200cc** also show relatively lower average ages, around 8.92 years and 9.08 years.
- **Royal Enfield Bullet Electra 350cc** has the highest average age among this group, at 13.79 years, followed by **Bajaj Pulsar 180cc** at 11.78 years.

This chart reveals that the average age of bikes varies significantly across different models. Newer, more recently introduced models like Bajaj V15 and KTM RC tend to have lower average ages, as expected. Conversely, some long-standing or older models, such as the Royal Enfield Bullet Electra and Bajaj Pulsar 180cc, show higher average ages, indicating their continued presence and resale activity in the used market even after many years.

Average age by Location

This chart shows the average price of bikes across various locations(top 10).

- **Mumbai** has the lowest average age among the displayed cities, at 8.67 years.
- **Hyderabad and Bangalore** also show relatively lower average ages, around 9.39 years and 9.83 years respectively.
- Cities like **Ahmedabad and Chennai** exhibit higher average ages, at 10.91 years and 10.47 years respectively.

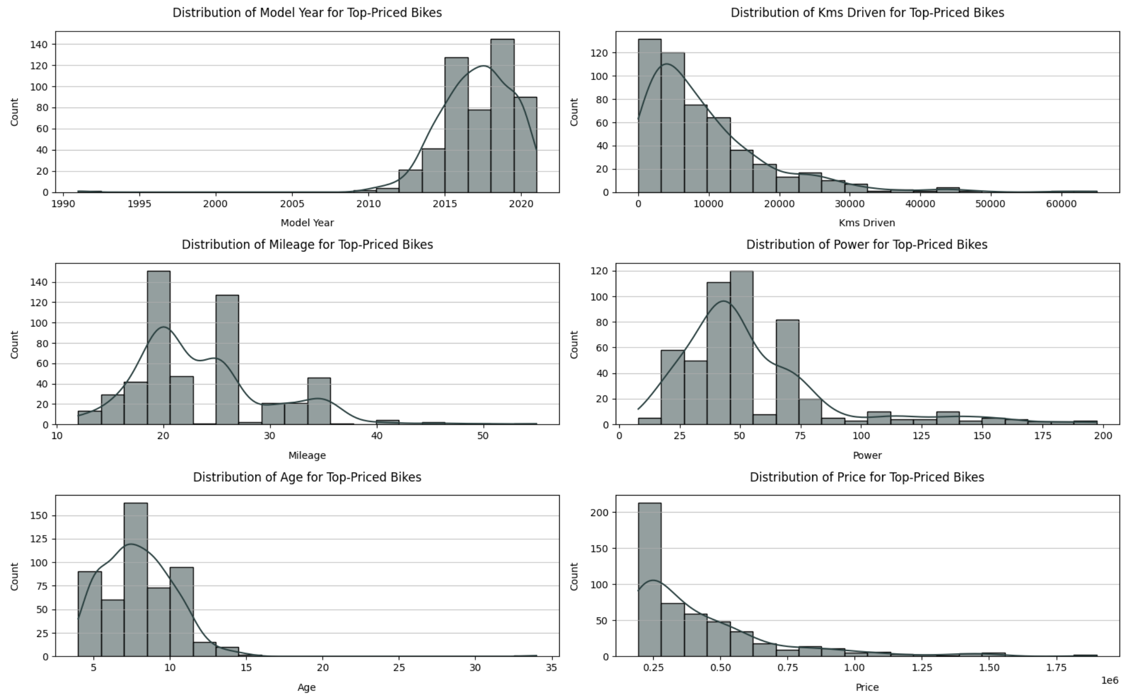
This chart highlights geographical differences in the average age of bikes listed. Major metropolitan areas like Mumbai and Hyderabad tend to have lower average ages, suggesting a more dynamic market with newer bikes being traded more frequently. In contrast, cities like Ahmedabad and Chennai show higher average ages, which could indicate that older bikes remain in circulation longer or are traded more in these regions.

1.4.15 Top-Priced(10%) Bikes Key Matrics Distribution

[126]:
```python
# Calculate top 10% bikes price
price_threshold = df['price'].quantile(0.90)

# Create new datafream for top 10% bikes
df_top_price = df[df['price'] >= price_threshold].copy()

print(f"\nAnalyzing bikes with prices >= {price_threshold:.2f} (Top 10% of␣
↪prices).")
print(f"Number of bikes in top 10% price category: {len(df_top_price)}")

print("\n" + "="*50 + "\n")

# Calculate the average of Key matrics
average = df_top_price[['model_year',
                        'kms_driven',
                        'mileage',
                        'power',
                        'age',
                        'price']].mean().reset_index().rename(columns={'index':␣
↪'Key Matrics',0:'Average'}).round(2)

# Calculate the STD of Key matrics
std = df_top_price[['model_year',
                    'kms_driven',
                    'mileage',
                    'power',
                    'age',
                    'price']].std().reset_index().rename(columns={'index': 'Key␣
↪Matrics',0:'Standard Deviation'}).round(2)

print("Key matrics average and atandard deviation of bikes with top prices:\n")
print(pd.merge(average, std, on='Key Matrics', how='inner'))

print("\n" + "="*50 + "\n")

# List of numerical columns to analyze
Key_matrics = ['model_year', 'kms_driven', 'mileage', 'power', 'age', 'price']

# set chart style properties
plt.figure(figsize=(16, 10))

for i, feature in enumerate(features_to_plot):

    # Create subplot
    plt.subplot(3, 2, i + 1)
```

```
    # Create histplot
    sns.histplot(df_top_price[feature], kde=True, bins=20, color='#2b4141')

    # set histplot properties
    plt.title(f'Distribution of {feature.replace("_", " ").title()} for␣
 ↪Top-Priced Bikes',y=1.05)
    plt.xlabel(feature.replace("_", " ").title(),labelpad=10)
    plt.ylabel('Count',labelpad=10)
    plt.grid(axis='y', alpha=0.75)

plt.tight_layout()
plt.show()
```

```
Analyzing bikes with prices >= 195000.00 (Top 10% of prices).
Number of bikes in top 10% price category: 509

==================================================

Key matrics average and atandard deviation of bikes with top prices:

  Key Matrics    Average  Standard Deviation
0  model_year    2017.07                2.52
1  kms_driven    9390.93             8967.53
2     mileage      23.53                6.28
3       power      55.37               32.96
4         age       7.93                2.52
5       price  424205.86           280416.59

==================================================
```

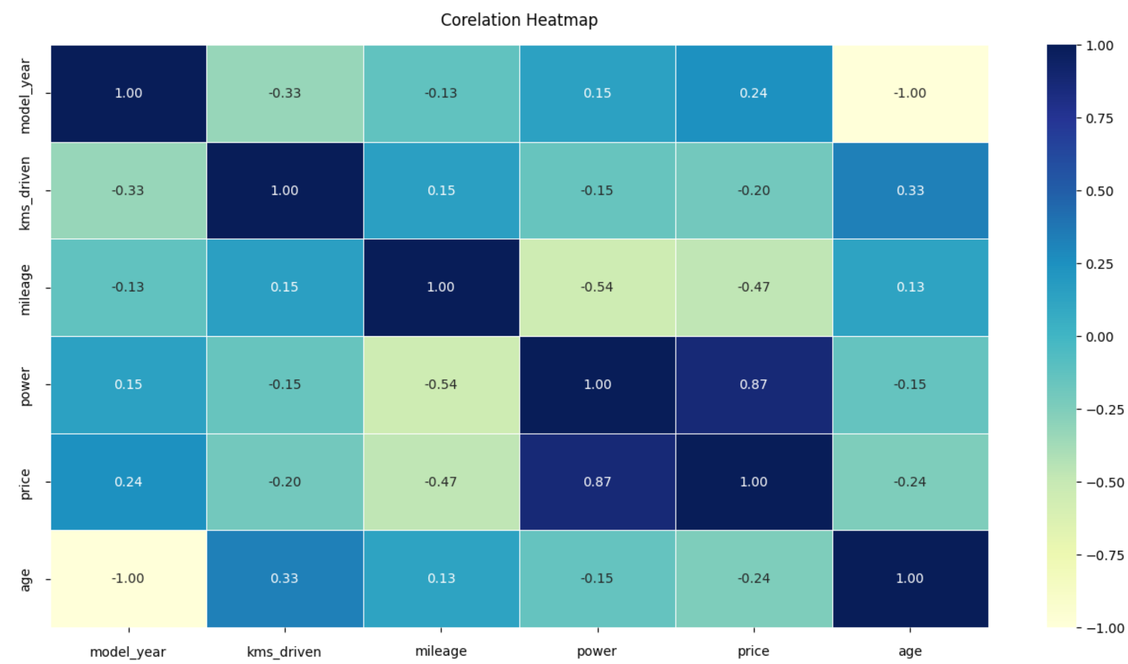


- **Model Year:** top-priced bikes are predominantly from newer model years, with a strong peak around 2015-2020. There are very few top-priced bikes from older model years like before 2010.
- **Kilometers Driven:** heavily right-skewed, indicating that the majority of top-priced bikes have very low kilometers driven, with a significant peak below 10,000 km. As Kms Driven increases, the count of top-priced bikes drops sharply.
- **Mileage:** shows a multi-modal or somewhat spread-out distribution. There's a notable peak at lower mileage values, and another smaller peak around 30-35 kmpl. This suggests that top-priced bikes aren't necessarily highly fuel-efficient; some high-performance bikes with lower mileage are also in this category.
- **Power:** is right-skewed, with a significant concentration of top-priced bikes having higher power outputs. This contrasts with the overall market distribution which is skewed towards lower power.
- **Age:** shows that top-priced bikes are predominantly younger, with a strong peak around 5-10 years of age. Very few top-priced bikes are older than 15 years. This is the inverse of the Model Year chart, as expected.
- **Price:** are concentrated at the higher end of the overall price spectrum. It's right-skewed within this top segment, indicating a range of high prices with fewer extremely high-priced bikes.

These charts collectively reveal the defining characteristics of bikes that command top prices in the used market. Such bikes are overwhelmingly newer and have very low kilometers driven, indicating minimal wear and tear. They also tend to possess higher engine power, suggesting a preference for performance and premium segments among high-value transactions. While mileage can vary, it's not necessarily a primary driver for top prices; instead, the focus shifts to modernity, low usage, and performance. This analysis clearly outlines the ideal profile of a used bike that fetches a premium.

1.4.16 Corelation Heatmap

[138]:
```
# set chart style properties
plt.figure(figsize=(16,8))

# Create Heatmap
ax = sns.heatmap(df.corr(numeric_only=True),
            annot=True,
            fmt='.2f',
            linewidths=.5,
            cmap="YlGnBu")

# set Heatmap properties
plt.title('Corelation Heatmap', y=1.02)
ax.tick_params(axis='both',pad=10)
plt.show()
```



This correlation heatmap provides crucial insights into the linear relationships between numerical features. It clearly shows that engine power (power) is the strongest positive predictor of a bike's price, with a very high correlation of 0.87. Model_year also positively influences price (0.24), while age (being inversely related to model_year) negatively impacts it. Interestingly, mileage has a notable negative correlation with price (-0.47), suggesting that higher-priced bikes often prioritize performance over fuel efficiency. Kms_driven shows a weaker negative correlation with price. The strong inverse relationship between power and mileage (-0.54) highlights the inherent trade-off between performance and fuel economy in bikes.

1.5 Recommendations

For Buyers

- Prioritize Ownership History: If budget allows, prioritize first owner bikes as they generally offer better value and are perceived to be better maintained.
- Newer bikes with lower Kilometers Driven will command higher prices but offer better reliability and modern features. For budget-conscious buyers, slightly older bikes with moderate Kilometers Driven can offer a good balance of value and usability.
- Research brands known for good resale value Royal Enfield, KTM etc. if long-term value retention is a concern. For pure utility, mass-market brands offer more affordable options.
- Understand your primary need. High-power bikes come at a premium and generally offer lower mileage. If fuel efficiency is key, focus on lower-power commuter segments.
- Be aware of regional price differences. It might be worthwhile to check listings in surrounding cities, as prices can vary.

For Sellers

- Bike Prices are competitively based on its brand, model name, age, Kilometers Driven, and owner type. Refer to market averages for similar bikes in your location.
- Emphasize low Kilometers Driven, recent model year, and high power as these features significantly drive up price.
- If possible, provide service history and maintenance records to justify the price, especially for bikes with higher Kilometers Driven or multiple owners.
- Ensure the bike is well-presented. Visual appeal can significantly impact perceived value.
- If you are in a smaller town, consider if listing in a nearby major city might fetch a better price, weighing the logistics.

[ ]:
```

```