Exploratory data analysis in Python.

Давайте разберемся, как исследовать данные в Python.

Введение

Что такое исследовательский анализ данных?

Исследовательский анализ данных или (EDA) исследует наборы данных, обобщая их основные характеристики, часто отображая их визуально. Этот шаг очень важен, особенно когда мы подходим к моделированию данных для применения машинного обучения. Графики в EDA состоят из гистограмм, прямоугольных диаграмм, точечных диаграмм и многого другого. На изучение данных часто уходит много времени (до 80% времени и сил из всего аналитического проекта). В процессе EDA мы можем определить формулировку проблемы (задачи) машинного обучения, что очень важно.

Как выполнить исследовательский анализ данных?

Это один из таких вопросов, на который каждый хочет знать ответ. Что ж, ответ в том, что это зависит от набора данных, с которым вы работаете. Не существует одного метода или общих методов для выполнения EDA, тогда как в этом руководстве вы можете

понять некоторые общие методы и графики, которые будут использоваться в процессе EDA.

Какие данные мы исследуем? Так как я большой поклонник автомобилей, я получил очень красивый набор данных автомобилей от Kaggle. Набор данных можно скачать здесь здесь. Чтобы дать краткую информацию о наборе данных, эти данные содержат более 10 000 строк и более 10 столбцов, которые содержат характеристики автомобиля, такие как Тип топлива двигателя, Мощность двигателя, Тип трансмиссии, MPG на шоссе, MPG в городе и другие. В этом руководстве мы исследуем данные и подготовим их для моделирования.

---

1. Импорт необходимых библиотек для EDA

Ниже приведены библиотеки, которые используются для выполнения EDA (исследовательского анализа данных) в этом руководстве.

```
import pandas as pd
import numpy as np
import seaborn as sns                          #visualisation
import matplotlib.pyplot as plt                #visualisation
%matplotlib inline
sns.set(color_codes=True)
```

---

2. Загрузка данных во data frame.

Загрузка данных в pandas data frame, безусловно, является одним из самых важных шагов в EDA, поскольку мы видим, что значение из набора данных разделено запятыми. Итак, все, что нам нужно сделать, это просто прочитать CSV в data frame, и pandas data frame сделает всю работу за нас.

Чтобы получить или загрузить набор данных в записную книжку, все, что я сделал, - это один тривиальный шаг. В Google Colab в левой части записной книжки вы найдете значок папки. Просто нужно перетащить файлы в эту область. Затем вы можете легко загрузить свой файл с помощью опции «Загрузить». Нет необходимости подключаться к диску Google или использовать какие-либо конкретные библиотеки, просто загрузите

набор данных, и ваша работа будет выполнена. На этом этапе следует помнить одну вещь: загруженные файлы будут удалены при повторном использовании этой среды выполнения. Вот так я получил набор данных в записную книжку.

```
df = pd.read_csv("data.csv")
# To display the top 5 rows
df.head(5)
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Numb Doo
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2

```
df.tail(5)                    # To display the botton 5 rows
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_W
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all whee
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all whee
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all whee
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all whee

3. Проверка типов данных (Checking the types of data)

Здесь мы проверяем типы данных, потому что иногда MSRP или цена автомобиля будут храниться в виде строки, и если в этом случае нам нужно преобразовать эту строку в целочисленные данные, только тогда мы сможем построить данные в виде графика. Здесь, в этом случае, данные уже в целочисленном формате, так что беспокоиться не о чем.

```
df.dtypes
```

```
Make                  object
Model                 object
Year                   int64
Engine Fuel Type      object
Engine HP            float64
Engine Cylinders     float64
Transmission Type     object
Driven_Wheels         object
Number of Doors      float64
Market Category       object
Vehicle Size          object
Vehicle Style         object
highway MPG            int64
city mpg               int64
Popularity             int64
MSRP                   int64
dtype: object
```

---

4. Удаление нерелевантных столбцов (Dropping irrelevant columns)

Этот шаг, безусловно, необходим в каждом EDA, потому что иногда может быть много столбцов, которые мы никогда не используем, в таких случаях удаление - единственное решение. В этом случае столбцы, такие как Тип топлива двигателя, Категория рынка, Стиль автомобиля, Популярность, Количество дверей, Размер транспортного средства, не имеют для меня никакого смысла, поэтому я просто отказался от них.

```
df = df.drop(['Engine Fuel Type', 'Market Category', 'Vehicle Style', 'Popularity', 'Numbe
df.head(5)
```

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	high
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	

5. Переименование столбцов (Renaming the columns)

В этом случае большинство имен столбцов очень сложно читать, поэтому просто изменим их имена. Это хороший подход, он улучшает читаемость набора данных (data set).

```
df = df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Cylinders", "Transmission
df.head(5)
```

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18

6. Удаление повторяющихся строк (Dropping the duplicate rows)

Это часто бывает удобно, потому что в наборе данных, который в нашем случае содержит более 10 000 строк, часто есть некоторые повторяющиеся данные, поэтому здесь удалим все повторяющиеся значения из набора данных. Например, до удаления у меня было 11914 строк данных, но после удаления дубликатов 10925 данных, что означает, что у меня было 989 повторяющихся данных.

```
df.shape
```

```
(11914, 10)
```

```
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (989, 10)
```

Теперь давайте удалим дубликаты.

```
df.count()      # Used to count the number of rows
```

```
Make            11914
Model           11914
Year            11914
HP              11845
Cylinders       11884
Transmission    11914
Drive Mode      11914
MPG-H           11914
MPG-C           11914
Price           11914
dtype: int64
```

Как видно выше, имеется 11914 строк, и мы удаляем 989 строк с повторяющимися данными.

```
df = df.drop_duplicates()
df.head(5)
```

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18

```
df.count()
```

```
Make            10925
Model           10925
Year            10925
HP              10856
Cylinders       10895
Transmission    10925
Drive Mode      10925
MPG-H           10925
MPG-C           10925
Price           10925
dtype: int64
```

---

7. Удаление отсутствующих или нулевых значений (Dropping the missing or null values)

Это в основном похоже на предыдущий шаг, но здесь все пропущенные значения обнаруживаются и удаляются позже. Это не лучший подход, потому что многие люди просто заменяют отсутствующие значения средним/медианой или модой для этого столбца, но в нашем случае просто удалим отсутствующие значения (так как их не много).

```
print(df.isnull().sum())
```

```
Make             0
Model            0
Year             0
HP              69
Cylinders       30
Transmission     0
Drive Mode       0
MPG-H            0
MPG-C            0
Price            0
dtype: int64
```

Это причина того, что на предыдущем шаге при подсчете числа значений в столбцах и Cylinders, и HP (лошадиных сил) было 10856 и 10895 из 10925 строк.

```
df = df.dropna()    # Dropping the missing values.
df.count()
```

```
Make            10827
Model           10827
Year            10827
HP              10827
Cylinders       10827
Transmission    10827
Drive Mode      10827
MPG-H           10827
MPG-C           10827
Price           10827
dtype: int64
```

Теперь мы удалили все строки, содержащие значения Null или N/A. (Cylinders and Horsepower (HP)).

```
print(df.isnull().sum())   # After dropping the values
```

```
Make              0
Model             0
Year              0
HP                0
Cylinders         0
Transmission      0
Drive Mode        0
MPG-H             0
MPG-C             0
Price             0
dtype: int64
```

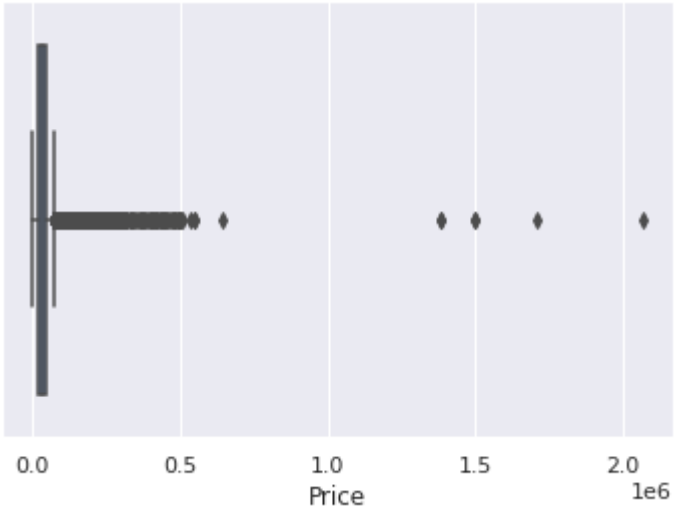
---

8. Обнаружение выбросов (Detecting Outliers)

Выброс - это точка или набор точек, которые отличаются от других точек. Часто бывает полезно обнаружить и удалить выбросы. Потому что выбросы являются одной из основных причин получения менее точной модели. Следовательно, их рекомендуется удалить. Обнаружение и удаление выбросов, которое мы собираемся выполнить, называется методом оценки IQR. Часто выбросы можно увидеть с помощью визуализаций, использующих box plot. Ниже показана диаграмма MSRP, Cylinders, Horsepower и EngineSize. На всех графиках вы можете обнаружить, что некоторые точки находятся за границами "усов", они не что иное, как выбросы. Техника поиска и удаления выбросов, которую мы выполняем в этом задании, взята из учебного руководства от towards data science.

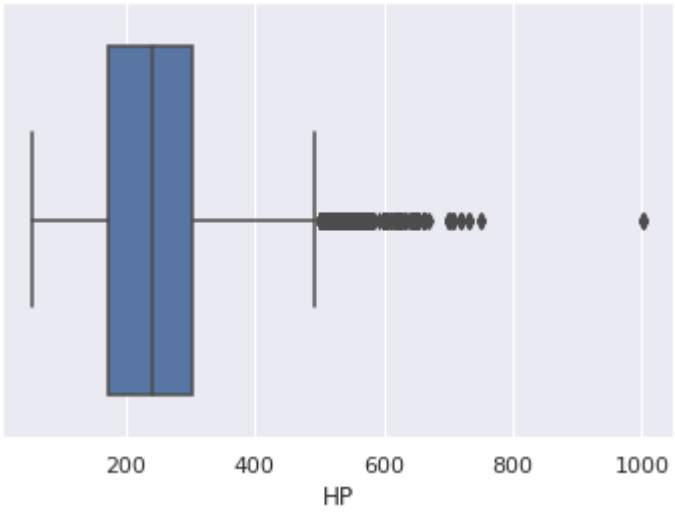
```
sns.boxplot(x=df['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc63ac19650>
```



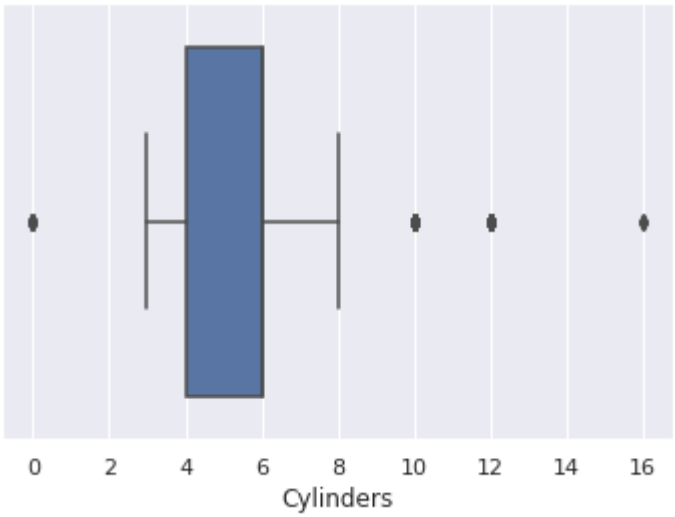

```
sns.boxplot(x=df['HP'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc63a343110>
```




```
sns.boxplot(x=df['Cylinders'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc639e8c490>
```




```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Year             9.0
HP             130.0
Cylinders        2.0
MPG-H            8.0
MPG-C            6.0
Price        21327.5
dtype: float64
```

Будем использовать этот метод, чтобы удалить выбросы.

```
df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape
```

```
(9191, 10)
```

Как видно выше, было удалено около 1600 строк. Но мы не можем полностью удалить выбросы, потому что даже после того, как вы воспользуетесь описанной выше техникой, возможно, останется 1–2 выброса, но это нормально, потому что выбросов было более 100. Так что это лучше, чем ничего.

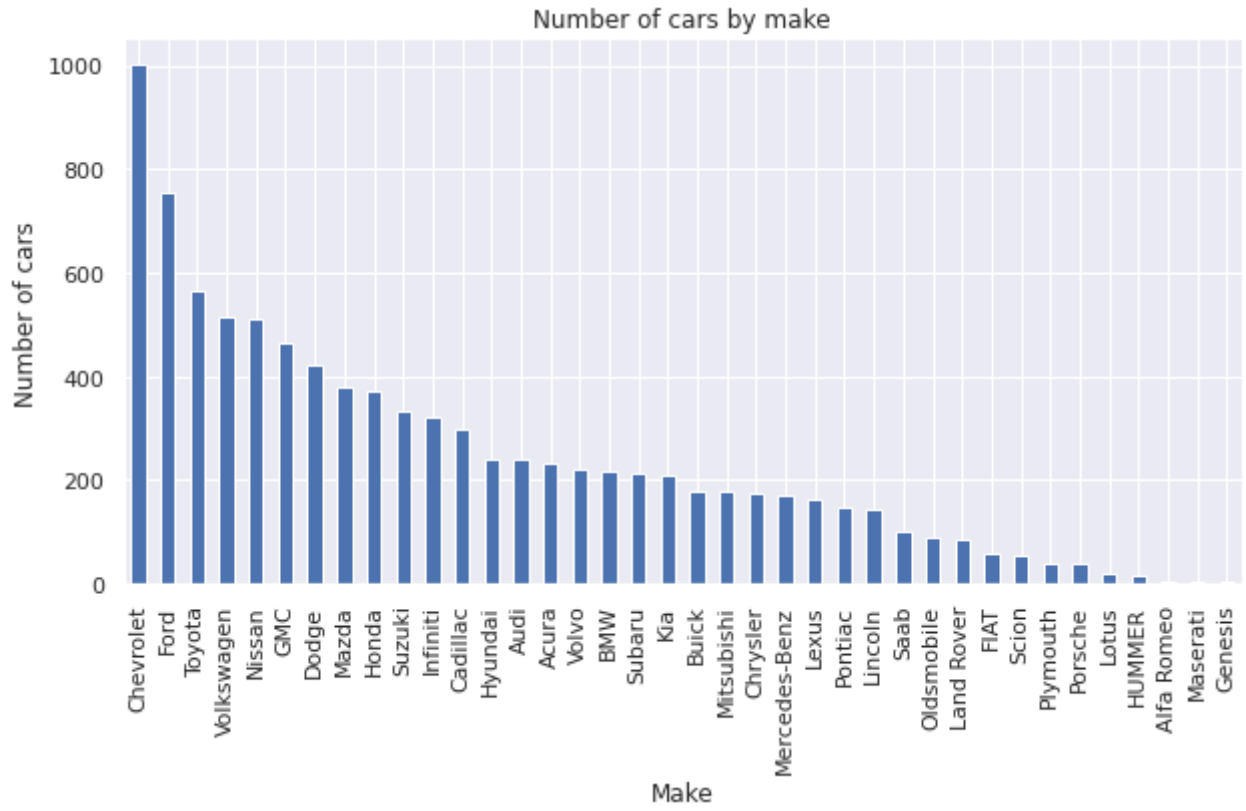
---

9. Построить график зависимости различных характеристик друг от друга (разброс) и частоты (гистограмма).

Histogram

Гистограмма отображает частоту появления переменных в определенном интервале. В нашем случае существует в основном 10 различных типов компаний-производителей автомобилей, но часто важно знать, у кого больше всего автомобилей. Создание этой гистограммы - тривиальное решение, которое позволяет нам узнать общее количество автомобилей, выпущенных разными компаниями.

```
df.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
```

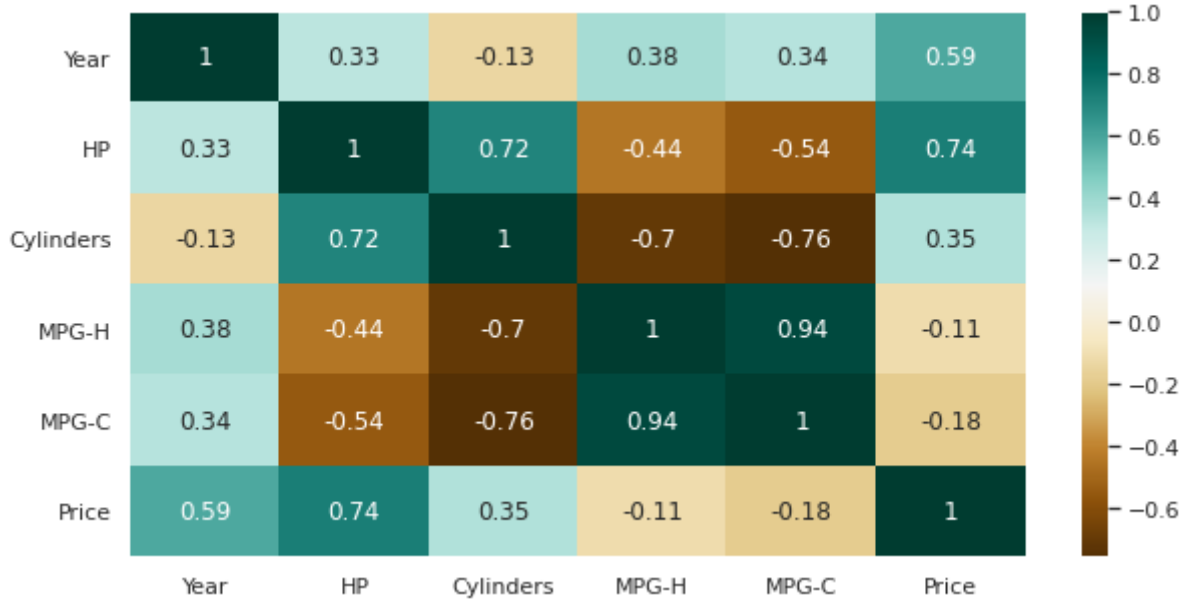


Heat Maps

Heat Maps - это тип визуализации, применяемый, когда нам нужно найти зависимые переменные. Один из лучших способов найти взаимосвязь между функциями - это использовать тепловые карты. На приведенной ниже Heat Maps мы знаем, что признак (фича) price в основном зависит от Engine Size, Horsepower, и Cylinders.

```
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c
```

	Year	HP	Cylinders	MPG-H	MPG-C	Price
Year	1.000000	0.326726	-0.133920	0.378479	0.338145	0.592983
HP	0.326726	1.000000	0.715237	-0.443807	-0.544551	0.739042
Cylinders	-0.133920	0.715237	1.000000	-0.703856	-0.755540	0.354013
MPG-H	0.378479	-0.443807	-0.703856	1.000000	0.939141	-0.106320
MPG-C	0.338145	-0.544551	-0.755540	0.939141	1.000000	-0.180515
Price	0.592983	0.739042	0.354013	-0.106320	-0.180515	1.000000

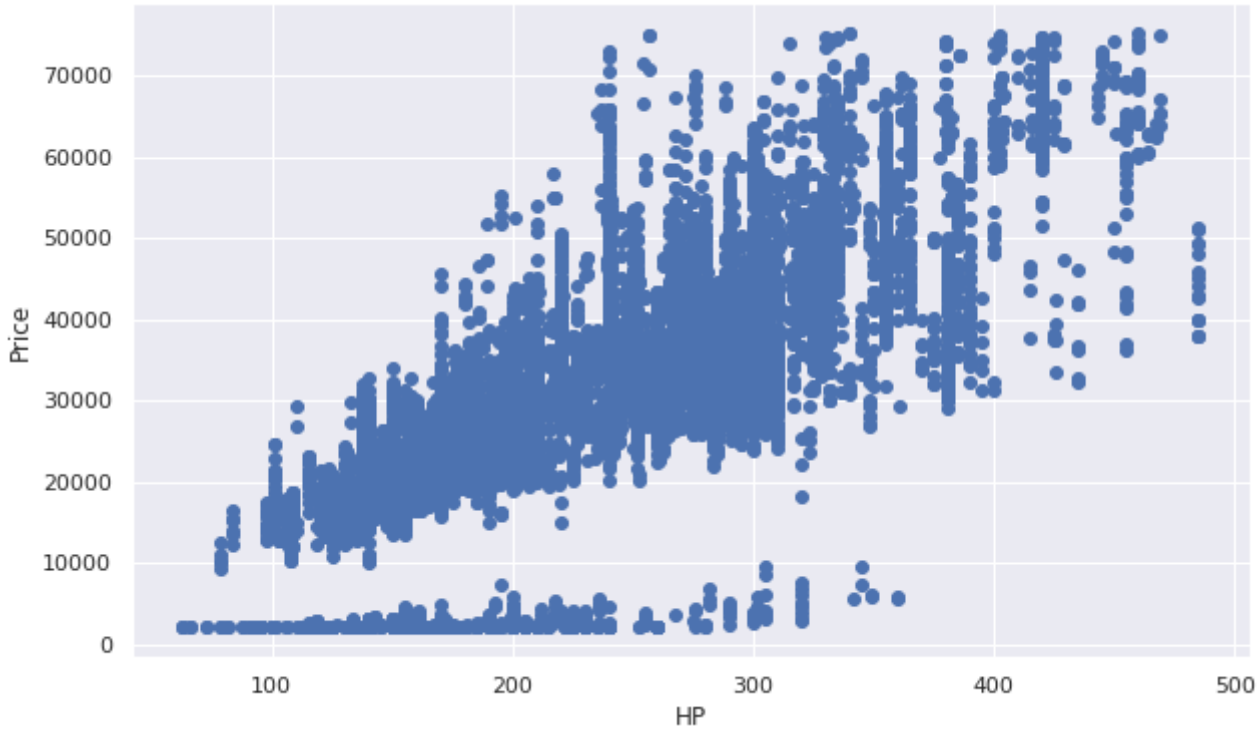


Scatterplot

Обычно мы используем диаграммы разброса, чтобы найти корреляцию (взаимосвязь) между двумя переменными. Здесь scatter plot построены для Horsepower и Price. С помощью графика, приведенного ниже, мы можем легко провести линию тренда.

```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(df['HP'], df['Price'])
ax.set_xlabel('HP')
```

```
ax.set_ylabel('Price')
plt.show()
```



Итак мы рассмотрели некоторые этапы исследовательского анализа данных, которые нужно выполнить в EDA.

Спасибо.