औद्योगिक प्रशिक्षण के लिए राष्ट्रीय संस्थान

National Institute for Industrial Training

One Premier Organization with Non Profit Status | Registered Under Govt. of WB
Empanelled Under Planning Commission Govt. of India
Inspired By: National Task Force on IT & SD Government of India

National Institute for Industrial Training- One Premier Organization with Non Profit Status Registered Under Govt.of West Bengal,Empanelled Under Planning Commission Govt.of India , Empanelled Under Central Social Welfare Board Govt. of India , Registered with National Career Services , Registered with National Employment Services.

SUBJECT: PYTHON DEVELOPER INTERNSHIP

SUBMITTED BY: DEBANJAN SAHA

SUBMITTED TO: Mr. SOUMOTANU MAZUMDAR

SUBMITTED ON: 11/12/2021

CONTENTS

- Acknowledgement
- Introduction
- Objectives
- Hardware and Software
- Code
- Snapshots
- Advantages
- Future Scope
- Conclusion
- Bibliography

- # STUDENT PROFILE:

- **NAME: DEBANJAN SAHA**
- **COLLEGE: UEM KOLKATA**
- **COURSE: BTECH (CSE)**
- **SEMESTER: 5RD SEMESTER (3RD YEAR)**
- **PHONE: +91 9475951336**
- **EMAIL: sahadebanjan9433@gmail.com**
- **YEAR OF PASSING: 2023**

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. I am highly indebted to **NATIONAL INSTITUTE FOR INDUSTRIAL TRAINING** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I would like to express my gratitude towards **Mr. SOUM0TANU MAZUMDAR** for his support, co-operation and encouragement which helped me for completion of this project.

INTRODUCTION

Python is a widely used general-purpose, high-level programming language. It was initially designed by GUIDO VAN ROSSUM in 1991 and developed by PYTHON SOFTWARE FOUNDATION. Python works on different platforms (Windows, Linux, Mac, Raspberry Pi, etc).

Python comes with a huge amount of inbuilt libraries. Many of the libraries are for Artificial Intelligence and Machine Learning. Some of the libraries are Tensorflow (which is high-level neural network library), scikit-learn (for data mining, data analysis and machine learning), pylearn2 (more flexible than scikit-learn), etc.

Python has an easy implementation for OpenCV. For other languages, students and researchers need to get to know the languages before getting into machine learning with that language. This is not the case with python. Even a programmer with very basic knowledge can easily handle python.

OBJECTIVES

1. Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics used for general-purpose programming.

2. Python also provides plenty of data mining tools that help in better handling the data.

3. Python is important for data scientists because it provides a vast variety of applications used in data science.

4. It also provides more flexibility in the field of Machine Learning and Deep Learning.

5. Python enables you to perform data analysis, data manipulation, and data visualization, which are very important in data science.

HARDWARE & SOFTWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

Operating system: Windows

Front End: Python 3.7

Platform: Anaconda Navigator

HARDWARE REQUIREMENTS

Machine:

Speed: 1.60 GHz & above

RAM: 8 GB

Hard disk: 1 TB HDD

CODE

LINEAR REGRESSION

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn import metrics

car_dataset = pd.read_csv("Car_Price.csv")

car_dataset.head()

car_dataset.shape

(205, 26)
car_dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
```

```
 4   aspiration       205 non-null    object
 5   doornumber       205 non-null    object
 6   carbody          205 non-null    object
 7   drivewheel       205 non-null    object
 8   enginelocation   205 non-null    object
 9   wheelbase        205 non-null    float64
 10  carlength        205 non-null    float64
 11  carwidth         205 non-null    float64
 12  carheight        205 non-null    float64
 13  curbweight       205 non-null    int64
 14  enginetype       205 non-null    object
 15  cylindernumber   205 non-null    object
 16  enginesize       205 non-null    int64
 17  fuelsystem       205 non-null    object
 18  boreratio        205 non-null    float64
 19  stroke           205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower       205 non-null    int64
 22  peakrpm          205 non-null    int64
 23  citympg          205 non-null    int64
 24  highwaympg       205 non-null    int64
 25  price            205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
car_dataset.describe()

car_dataset.CarName.unique()

array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw
x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega
2300',
```

'dodge rampage', 'dodge challenger se', 'dodge d200', 'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)', 'dodge coronet custom', 'dodge dart custom', 'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc', 'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl', 'honda accord', 'honda civic 1300', 'honda prelude', 'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ', 'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk', 'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4', 'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs', 'mazda glc 4', 'mazda glc custom l', 'mazda glc custom', 'buick electra 225 custom', 'buick century luxus (sw)', 'buick century', 'buick skyhawk', 'buick opel isuzu deluxe', 'buick skylark', 'buick century special', 'buick regal sport coupe (turbo)', 'mercury cougar', 'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander', 'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero', 'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue', 'nissan latio', 'nissan titan', 'nissan leaf', 'nissan juke', 'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz', 'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks', 'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl', 'peugeot 505s turbo diesel', 'plymouth fury iii', 'plymouth cricket', 'plymouth satellite custom (sw)', 'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster', 'porsche macan', 'porcshce panamera', 'porsche cayenne', 'porsche boxter', 'renault 12tl', 'renault 5 gtl', 'saab 99e', 'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz', 'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',

```
       'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
       'toyota corolla 1200', 'toyota corona hardtop',
       'toyota corolla 1600 (sw)', 'toyota carina', 'toyota mark ii',
       'toyota corolla', 'toyota corolla liftback',
       'toyota celica gt liftback', 'toyota corolla tercel',
       'toyota corona liftback', 'toyota starlet', 'toyota tercel',
       'toyota cressida', 'toyota celica gt', 'toyouta tercel',
       'vokswagen rabbit', 'volkswagen 1131 deluxe sedan',
       'volkswagen model 111', 'volkswagen type 3', 'volkswagen
411 (sw)',
       'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
       'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
       'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
       'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
car_dataset.price.sum()

2721725.667
car_dataset.carbody.unique()

array(['convertible', 'hatchback', 'sedan', 'wagon', 'hardtop'],
      dtype=object)
car_dataset.isnull().sum()

car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
```

```
carheight         0
curbweight        0
enginetype        0
cylindernumber    0
enginesize        0
fuelsystem        0
boreratio         0
stroke            0
compressionratio  0
horsepower        0
peakrpm           0
citympg           0
highwaympg        0
price             0
dtype: int64
print(car_dataset.fueltype.value_counts())

print(car_dataset.carbody.value_counts())

gas      185
diesel    20
Name: fueltype, dtype: int64
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: carbody, dtype: int64

y=car_dataset[["price"]]

x=car_dataset[["enginesize"]]

lm = LinearRegression()

lm.fit(x_train,y_train)
```

```
print(lm.intercept_,lm.coef_)

[-7107.69540612] [[161.34743434]]
pred=lm.predict(x_test)

print(pred[0:10])

[[ 7413.57368461]
 [10479.1749371 ]
 [20482.71586628]
 [ 8704.35315935]
 [12253.99671486]
 [22096.19020969]
 [12415.3441492 ]
 [ 8543.005725  ]
 [10640.52237144]
 [ 7736.2685533 ]]
training_pred=lm.predict(x_test)

error_score=metrics.r2_score(y_test,training_pred)

print( error_score)

0.8258154601020361
plt.scatter(y_test,training_pred,color="red")

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.show()

sns.jointplot(y=car_dataset[["price"]],x=car_dataset[["enginesize",]],data=car_dataset,kind="reg",color="blue")

sns.distplot(y_test-pred,bins=50,color="red",kde=False)

plt.figure(figsize = (15, 10))
```

```
sns.heatmap(car_dataset.corr(), annot = True, cmap="YlGnBu")
plt.show()
plt.figure(figsize = (10,5))
sns.countplot(x="enginesize",data=car_dataset)
plt.show()
plt.figure(figsize = (10,5))
sns.violinplot(x="drivewheel",y="price",data=car_dataset)
plt.show()
plt.figure(figsize = (10,5))
sns.countplot(x="fuelsystem",data=car_dataset)
plt.show()
sns.boxplot(x="enginesize",data=car_dataset,color="green")
plt.figure(figsize = (5,5))
sns.stripplot(x="enginesize",y="price",data=car_dataset)
plt.show()
plt.figure(figsize = (5,5))
sns.kdeplot(car_dataset.price,car_dataset.enginesize)
plt.show()
df=
pd.DataFrame(car_dataset.groupby(['fueltype'])['price'].mean(
).sort_values(ascending = False))
```

```
df.plot.bar()
plt.title('Fuel Type vs Average Price')
plt.show()
```

Linear Regression

- **Importing Library For Linear Regression**

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

- **Importing Data set & Linear regression model For Linear Regression**

```
df=pd.read_csv("House_Price.csv")
df.head()
df.shape
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

- **Depending upon number of rooms, predicting the price of house**

```
y=df[["price"]]
```

```
x=df[["room_num"]]

lm = LinearRegression()

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.4, random_state=101)

lm.fit(x_train,y_train)
```

- **Finding intercept & coefficient**

```
print(lm.intercept_,lm.coef_)

[-31.71878837] [[8.54740782]]
```

- **Predicting the price**

```
pred=lm.predict(x_test)

print(pred[0:10])

[[35.59204817]
[29.36953528]
[21.34351934]
[22.13842827]
[24.65136617]
[30.13025458]
[42.85734481]
[19.13828813]
[27.62586409]
[15.55692425]]
```

- **Plotting the graphs for the linear regression model**

```
plt.scatter(y_test,pred,color="green")
plt.xlabel("price")
plt.ylabel("room_num")
plt.title("Scatter Plot")

sns.distplot((y_test-pred),bins=50,color="blue",kde= False)
plt.xlabel("price")
plt.ylabel("room_num")
plt.title("Histogram")

sns.jointplot(x=df[["price"]],y=df[["room_num"]],data=df,kind="reg",color="magenta")
plt.xlabel("price")
plt.ylabel("room_num")
```

- **Finding error**

```
from sklearn import metrics
metrics.mean_absolute_error(y_test,pred)
4.774097035805513
metrics.mean_squared_error(y_test,pred)
48.359147224858454

print(np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

6.954074145769403

K –Means Clustering

- **Importing the Library**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
```

- **PREPARING DATASET FOR K-MEANS CLUSTERING**

```
dataset = make_blobs(n_samples=200,
                     centers=4,
                     n_features=2,
                     cluster_std=1.6,
                     random_state=50)
points = dataset[0]
# Create a K means object
kmeans = KMeans(n_clusters=4)
# Fit this Kmeans object to this dataset
kmeans.fit(points)
```

- **PLOTTING THE CLUSTERS FOR THE PREPARED DATASET**

```
plt.scatter(dataset[0][:,0],dataset[0][:,1],c=dataset[1],cmap="rainbow")

clusters = kmeans.cluster_centers_

print(clusters)

[[-5.56465793 -2.34988939]
 [-2.40167949 10.17352695]
 [ 0.05161133 -5.35489826]
 [-1.92101646  5.21673484]]
```

- **CLUSTERS GRAPH WITHOUT CENTERS**

```
y_km = kmeans.fit_predict(points)

plt.scatter(points[y_km == 0,0],points[y_km == 0,1],s=50,color="red")

plt.scatter(points[y_km == 1,0],points[y_km == 1,1],s=50,color="blue")

plt.scatter(points[y_km == 2,0],points[y_km == 2,1],s=50,color="orange")

plt.scatter(points[y_km == 3,0],points[y_km == 3,1],s=50,color="magenta")
```

- **CLUSTERS GRAPH WITH CENTERS**

```
plt.scatter(points[y_km == 0,0],points[y_km == 0,1],s=50,color="red")

plt.scatter(points[y_km == 1,0],points[y_km == 1,1],s=50,color="cyan")
```

```
plt.scatter(points[y_km == 2,0],points[y_km == 2,1],s=50,color="orange")

plt.scatter(points[y_km == 3,0],points[y_km == 3,1],s=50,color="magenta")

plt.scatter(clusters[0][0],clusters[0][1],marker="*",s=500, color="black")

plt.scatter(clusters[1][0],clusters[1][1],marker="*",s=500, color="black")

plt.scatter(clusters[2][0],clusters[2][1],marker="*",s=500, color="black")

plt.scatter(clusters[3][0],clusters[3][1],marker="*",s=500, color="black")

# Making the Clusters

f, (ax1, ax2) = plt.subplots(nrows=1,
 ncols=2,
 sharey=True,
 figsize=(10,6))


ax1.scatter(dataset[0][:,0],
 dataset[0][:,1],
 c=y_km,
 cmap='rainbow')
```

```
ax2.scatter(dataset[0][:,0],
 dataset[0][:,1],
 c=dataset[1],
 cmap='rainbow')

ax1.scatter(x=clusters[:, 0],
 y=clusters[:, 1],
 c='black',
 s=300,
 alpha=0.5);
```

SNAPSHOT

LINEAR REGRESSION

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

```
car_dataset = pd.read_csv("Car_Price.csv")
```

```
car_dataset.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	stro
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3

5 rows × 26 columns

```
1 car_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
car_dataset.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepowe
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.00000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	10.142537	104.11707
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	3.972040	39.54416
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.00000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	8.600000	70.00000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.00000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	9.400000	116.00000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	288.00000

```
car_dataset.CarName.unique()
```

```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
       'dodge rampage', 'dodge challenger se', 'dodge d200',
       'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
       'dodge coronet custom', 'dodge dart custom',
       'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
       'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
       'honda accord', 'honda civic 1300', 'honda prelude',
       'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ',
       'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
       'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
       'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
```

```
1 car_dataset.price.sum()
```

```
2721725.667
```

```
1 car_dataset.carbody.unique()
```

```
array(['convertible', 'hatchback', 'sedan', 'wagon', 'hardtop'],
      dtype=object)
```

```
1 car_dataset.isnull().sum()
```

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
```

```
print(car_dataset.fueltype.value_counts())
print(car_dataset.carbody.value_counts())
```

```
gas       185
diesel     20
Name: fueltype, dtype: int64
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: carbody, dtype: int64
```

```
y=car_dataset[["price"]]
x=car_dataset[["enginesize"]]
```

```
lm = LinearRegression()
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

```
lm.fit(x_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
print(lm.intercept_,lm.coef_)
```

```
[-7107.69540612] [[161.34743434]]
```

```
pred=lm.predict(x_test)
print(pred[0:10])
```

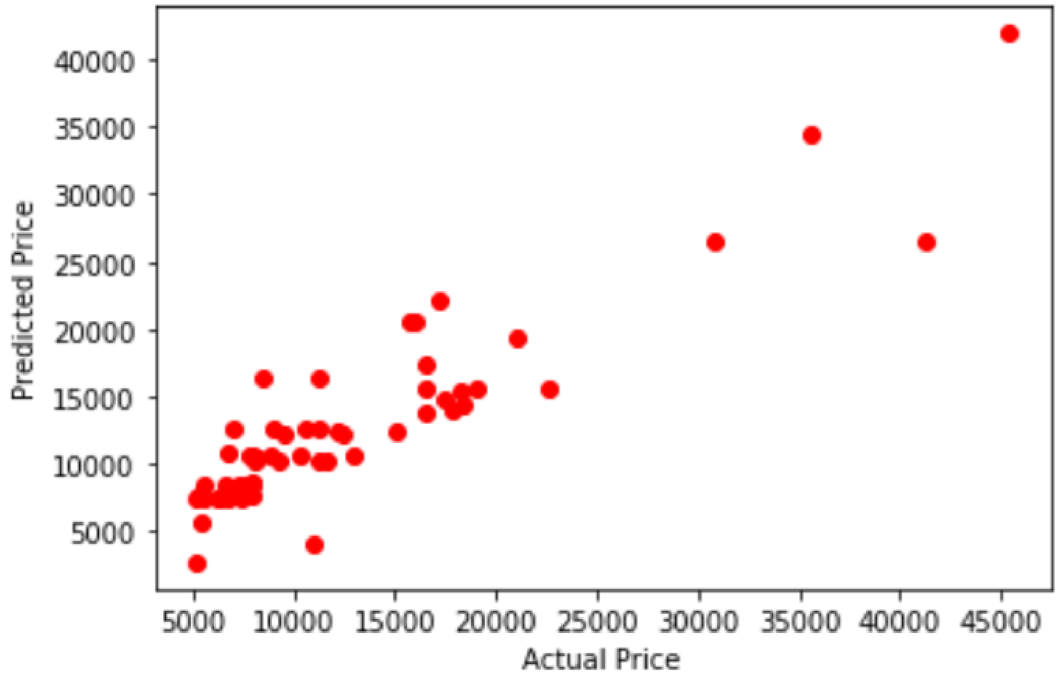
```
[[ 7413.57368461]
 [10479.1749371 ]
 [20482.71586628]
 [ 8704.35315935]
 [12253.99671486]
 [22096.19020969]
 [12415.3441492 ]
 [ 8543.005725  ]
 [10640.52237144]
 [ 7736.2685533 ]]
```

```
training_pred=lm.predict(x_test)
```

```
error_score=metrics.r2_score(y_test,training_pred)
print(error_score)
```

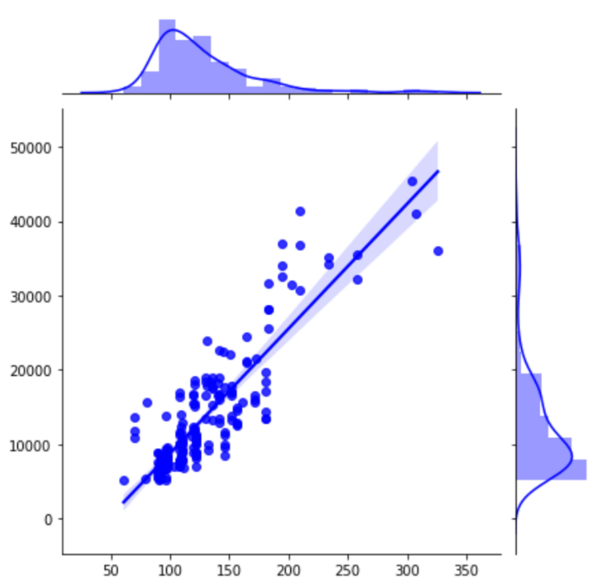
```
0.825815460102036
```

```python
plt.scatter(y_test,training_pred,color="red")
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.show()
```



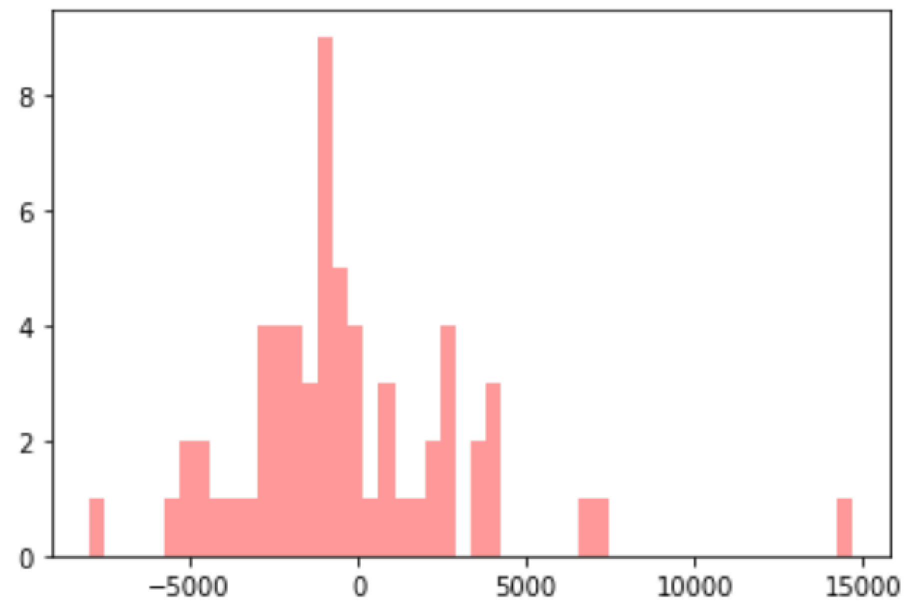

```python
sns.jointplot(y=car_dataset[["price"]],x=car_dataset[["enginesize",]],data=car_dataset,kind="reg",color="blue")
```

```
<seaborn.axisgrid.JointGrid at 0x2b4b0405d88>
```

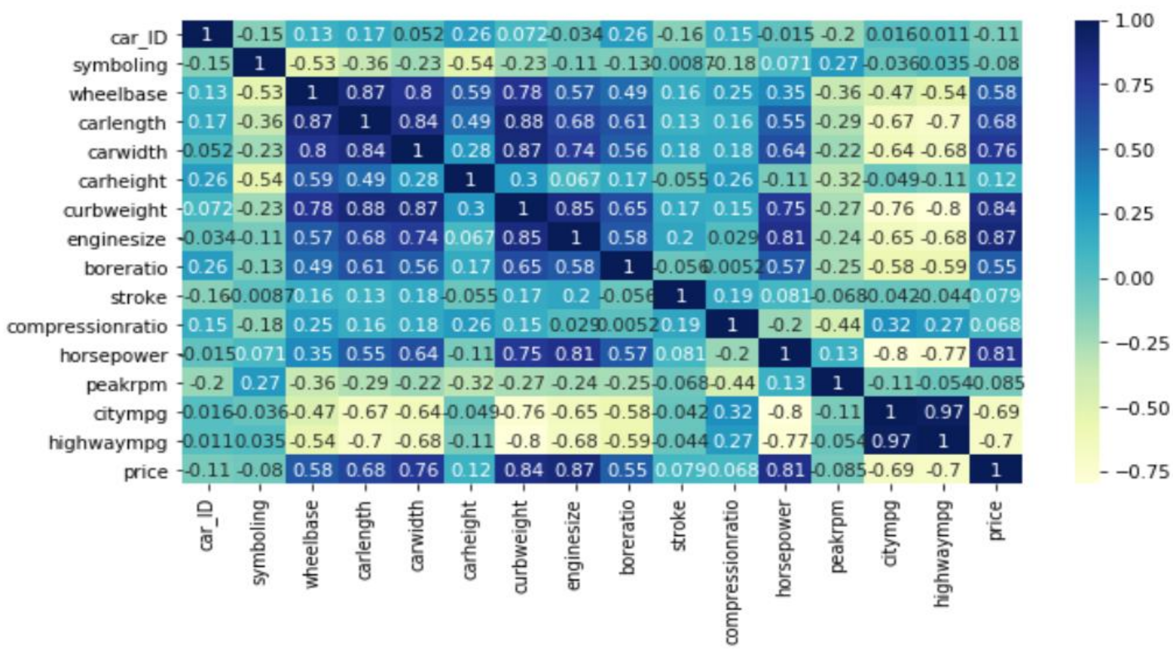


```python
sns.distplot(y_test-pred,bins=50,color="red",kde=False)
```

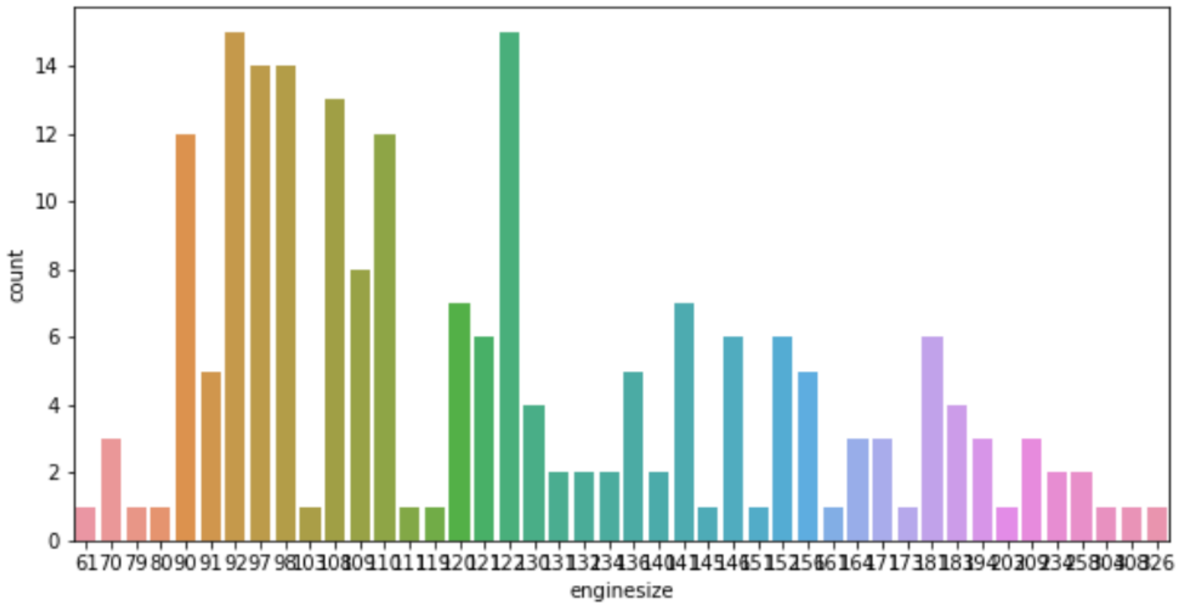
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b4b053a108>
```



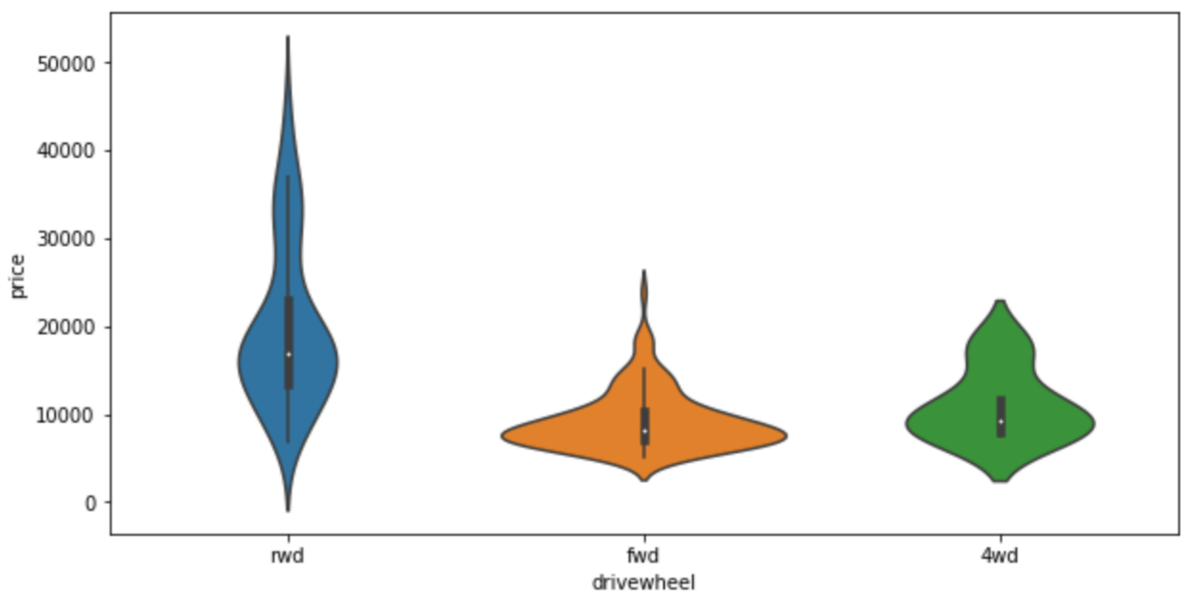

```python
plt.figure(figsize = (10, 5))
sns.heatmap(car_dataset.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



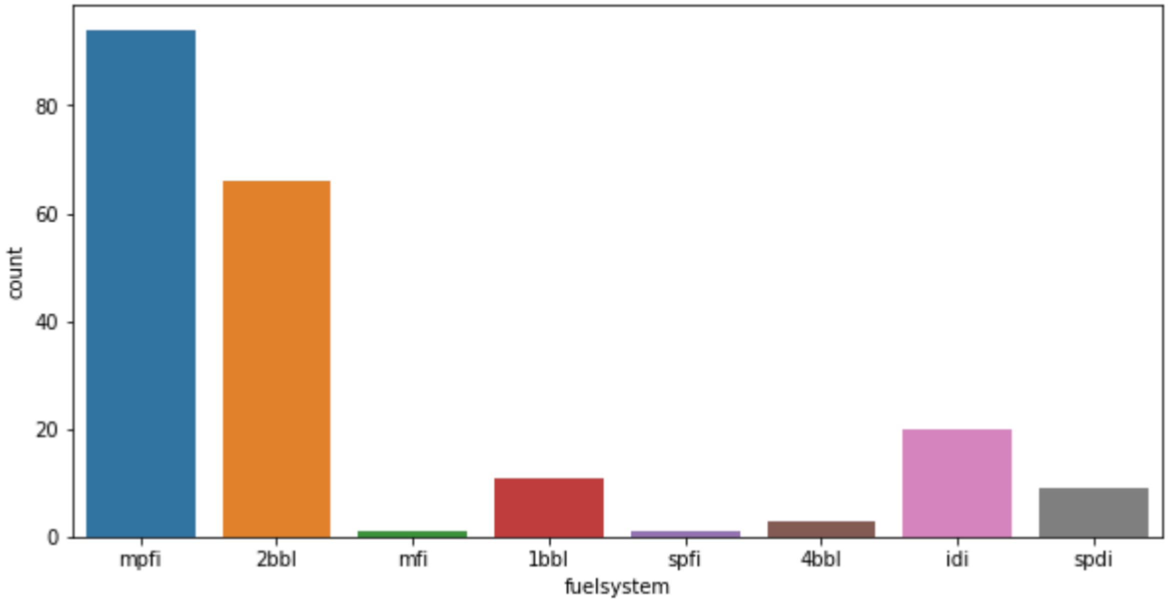
```python
plt.figure(figsize = (10,5))
sns.countplot(x="enginesize",data=car_dataset)
plt.show()
```




```python
plt.figure(figsize = (10,5))
sns.violinplot(x="drivewheel",y="price",data=car_dataset)
plt.show()
```

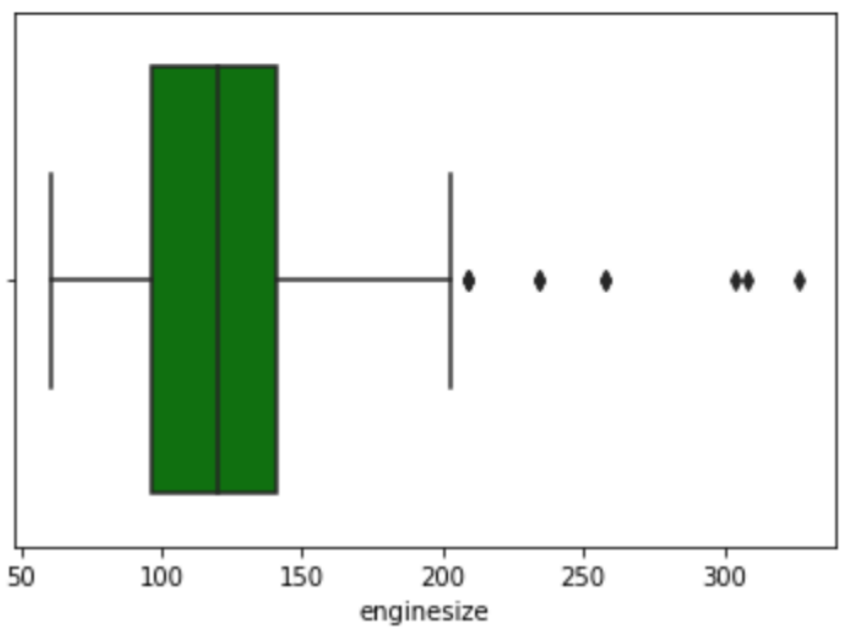


```
plt.figure(figsize = (10,5))
sns.countplot(x="fuelsystem",data=car_dataset)
plt.show()
```

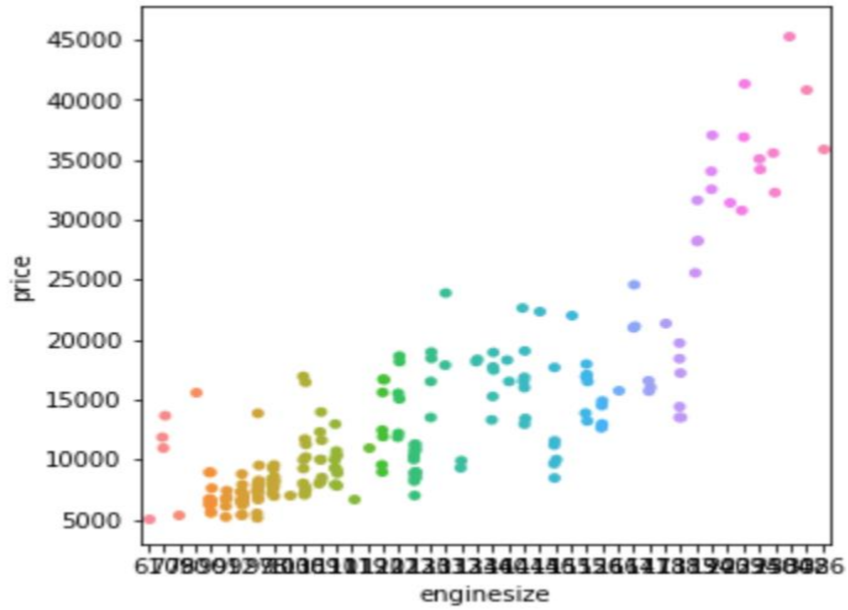



```
sns.boxplot(x="enginesize",data=car_dataset,color="green")
```

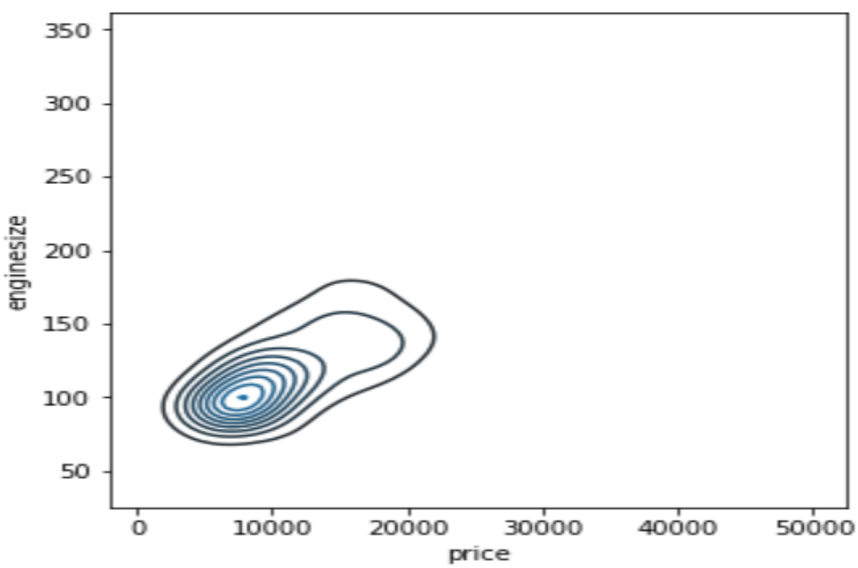
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b4b0967c08>
```



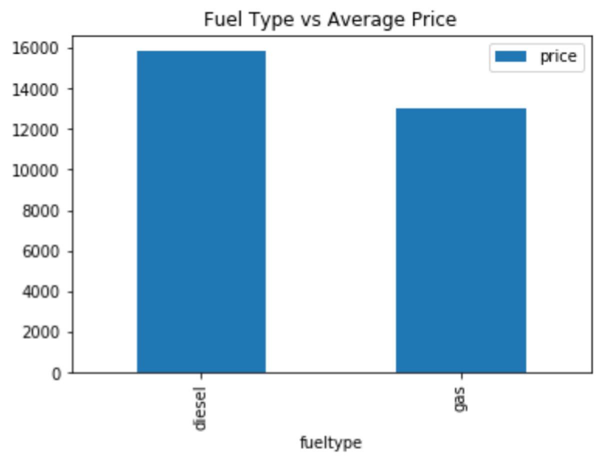
```
plt.figure(figsize = (5,5))
sns.stripplot(x="enginesize",y="price",data=car_dataset)
plt.show()
```




```
plt.figure(figsize = (5,5))
sns.kdeplot(car_dataset.price,car_dataset.enginesize)
plt.show()
```



```python
df= pd.DataFrame(car_dataset.groupby(['fueltype'])['price'].mean().sort_values(ascending = False))
df.plot.bar()
plt.title('Fuel Type vs Average Price')
plt.show()
```



LINEAR REGRESSION

CODE

- **IMPORTING LIBRARY FOR LINEAR REGRESSION**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

- **IMPORTING DATA SET & LINEAR REGRESSION MODEL FOR LINEAR REGRESSION**

```
df=pd.read_csv("House_Price.csv")
df.head()
df.shape
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

- **DEPENDING UPON NUMBER OF ROOMS, PREDICTING THE PRICE OF HOUSE**

```
y=df[["price"]]
x=df[["room_num"]]
lm = LinearRegression()
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.4, random_state=101)
lm.fit(x_train,y_train)
```

- **FINDING INTERCEPT & COEFFICIENT**

```
print(lm.intercept_,lm.coef_)
```

[-31.71878837] [[8.54740782]]

- **PREDICTING THE PRICE**

```
pred=lm.predict(x_test)

print(pred[0:10])
```

```
[[35.59204817]
[29.36953528]
[21.34351934]
[22.13842827]
[24.65136617]
[30.13025458]
[42.85734481]
[19.13828813]
[27.62586409]
[15.55692425]]
```

- **PLOTTING THE GRAPHS FOR THE LINEAR REGRESSION MODEL**

```
plt.scatter(y_test,pred,color="green")

plt.xlabel("price")

plt.ylabel("room_num")

plt.title("Scatter Plot")


sns.distplot((y_test-pred),bins=50,color="blue",kde=False)
```

plt.xlabel("price")

plt.ylabel("room_num")

plt.title("Histogram")

sns.jointplot(x=df[["price"]],y=df[["room_num"]],data =df,kind="reg",color="magenta")

plt.xlabel("price")

plt.ylabel("room_num")

- **FINDING ERROR**

from sklearn import metrics

metrics.mean_absolute_error(y_test,pred)

4.774097035805513

metrics.mean_squared_error(y_test,pred)

48.359147224858454

print(np.sqrt(metrics.mean_squared_error(y_test,pred)))

6.954074145769403

K –MEANS CLUSTERING

CODE

- **IMPORTING THE LIBRARY**

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import make_blobs

from sklearn.cluster import KMeans
```

- **PREPARING DATASET FOR K-MEANS CLUSTERING**

```
dataset = make_blobs(n_samples=200,
                     centers=4,
                     n_features=2,
                     cluster_std=1.6,
                     random_state=50)

points = dataset[0]

# Create a K means object
```

```
kmeans = KMeans(n_clusters=4)
# Fit this Kmeans object to this dataset
kmeans.fit(points)
```

- **PLOTTING THE CLUSTERS FOR THE PREPARED DATASET**

```
plt.scatter(dataset[0][:,0],dataset[0][:,1],c=dataset[1],cmap="rainbow")
clusters = kmeans.cluster_centers_
print(clusters)
```

```
[[-5.56465793 -2.34988939]
 [-2.40167949 10.17352695]
 [ 0.05161133 -5.35489826]
 [-1.92101646  5.21673484]]
```

- **CLUSTERS GRAPH WITHOUT CENTERS**

```
y_km = kmeans.fit_predict(points)
plt.scatter(points[y_km == 0,0],points[y_km == 0,1],s=50,color="red")
plt.scatter(points[y_km == 1,0],points[y_km == 1,1],s=50,color="blue")
plt.scatter(points[y_km == 2,0],points[y_km == 2,1],s=50,color="orange")
```

```
plt.scatter(points[y_km == 3,0],points[y_km == 3,1],s=50,color="magenta")
```

- **Clusters graph with centers**

```
plt.scatter(points[y_km == 0,0],points[y_km == 0,1],s=50,color="red")

plt.scatter(points[y_km == 1,0],points[y_km == 1,1],s=50,color="cyan")

plt.scatter(points[y_km == 2,0],points[y_km == 2,1],s=50,color="orange")

plt.scatter(points[y_km == 3,0],points[y_km == 3,1],s=50,color="magenta")

plt.scatter(clusters[0][0],clusters[0][1],marker="*", s=500,color="black")

plt.scatter(clusters[1][0],clusters[1][1],marker="*", s=500,color="black")

plt.scatter(clusters[2][0],clusters[2][1],marker="*", s=500,color="black")

plt.scatter(clusters[3][0],clusters[3][1],marker="*", s=500,color="black")

# Making the Clusters

f, (ax1, ax2) = plt.subplots(nrows=1,
```

```
 ncols=2,
 sharey=True,
 figsize=(10,6))

ax1.scatter(dataset[0][:,0],
 dataset[0][:,1],
 c=y_km,
 cmap='rainbow')

ax2.scatter(dataset[0][:,0],
 dataset[0][:,1],
 c=dataset[1],
 cmap='rainbow')

ax1.scatter(x=clusters[:, 0],
 y=clusters[:, 1],
 c='black',
 s=300,
 alpha=0.5);
```

In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```python
df=pd.read_csv("House_Price.csv")
```

In [3]:
```python
df.head()
```

Out[3]:

	price	crime_rate	resid_area	air_qual	room_num	age	dist1	dist2	dist3	dist4
0	24.0	0.00632	32.31	0.538	6.575	65.2	4.35	3.81	4.18	4.01
1	21.6	0.02731	37.07	0.469	6.421	78.9	4.99	4.70	5.12	5.06
2	34.7	0.02729	37.07	0.469	7.185	61.1	5.03	4.86	5.01	4.97
3	33.4	0.03237	32.18	0.458	6.998	45.8	6.21	5.93	6.16	5.96
4	36.2	0.06905	32.18	0.458	7.147	54.2	6.16	5.86	6.37	5.86

In [4]:
```python
df.shape
```

Out[4]:
```
(506, 20)
```

In [5]:
```python
from sklearn.linear_model import LinearRegression
```

In [6]:
```python
from sklearn.model_selection import train_test_split
```

In [7]:
```python
y=df[["price"]]
```

In [8]:
```
x=df[["room_num"]]
```

In [9]:
```
lm = LinearRegression()
```

In [10]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4, random_state=101)
```

In [11]:
```
lm.fit(x_train,y_train)
```

Out[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [12]:
```
print(lm.intercept_,lm.coef_)
```

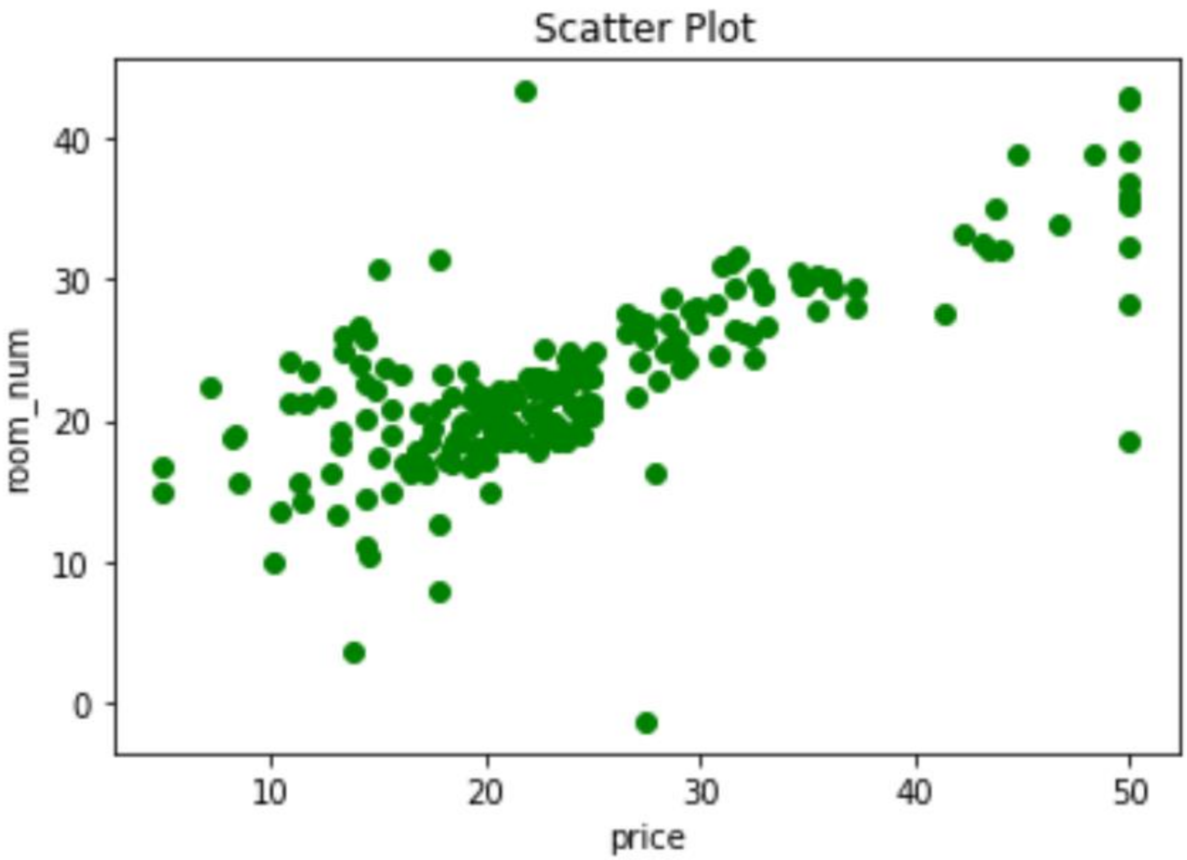
```
[-31.71878837] [[8.54740782]]
```

In [13]:
```
pred=lm.predict(x_test)
print(pred[0:10])
```

```
[[35.59204817]
 [29.36953528]
 [21.34351934]
 [22.13842827]
 [24.65136617]
 [30.13025458]
 [42.85734481]
 [19.13828813]
 [27.62586409]
 [15.55692425]]
```

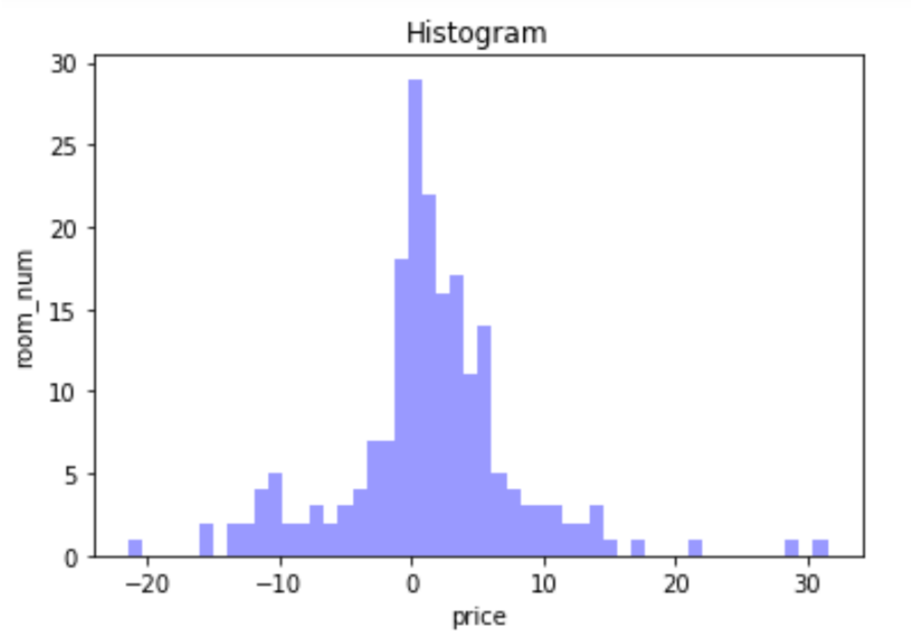
In [14]:
```
plt.scatter(y_test,pred,color="green")
plt.xlabel("price")
plt.ylabel("room_num")
plt.title("Scatter Plot")
```

Out[14]: `Text(0.5, 1.0, 'Scatter Plot')`



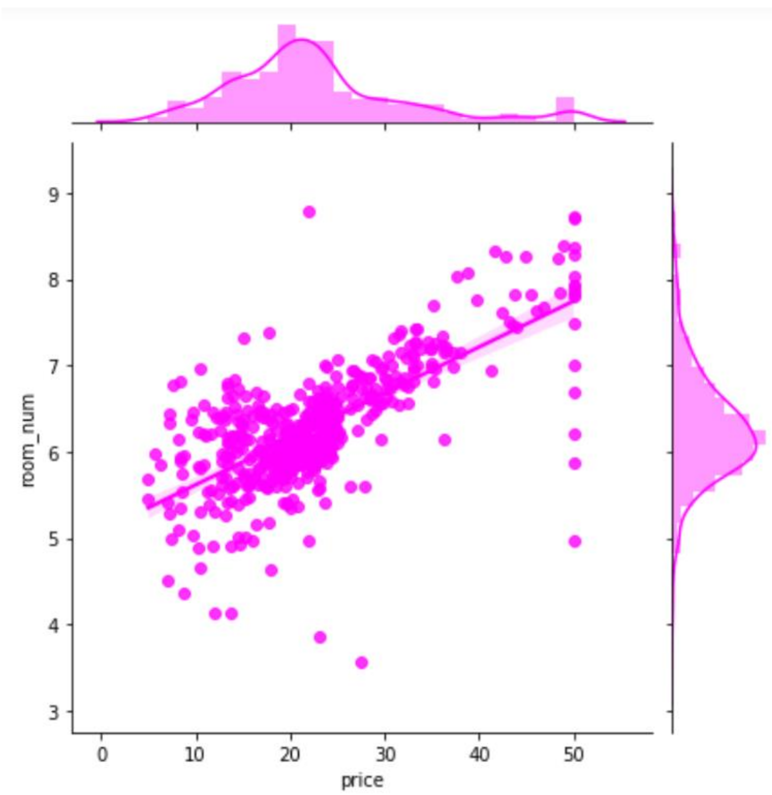
In [15]:
```
sns.distplot((y_test-pred),bins=50,color="blue",kde= False)
plt.xlabel("price")
plt.ylabel("room_num")
plt.title("Histogram")
```

Out[15]: `Text(0.5, 1.0, 'Histogram')`




```
In [16]: sns.jointplot(x=df[["price"]],y=df[["room_num"]],data=df,kind="reg",color="magenta")
         plt.xlabel("price")
         plt.ylabel("room_num")
```

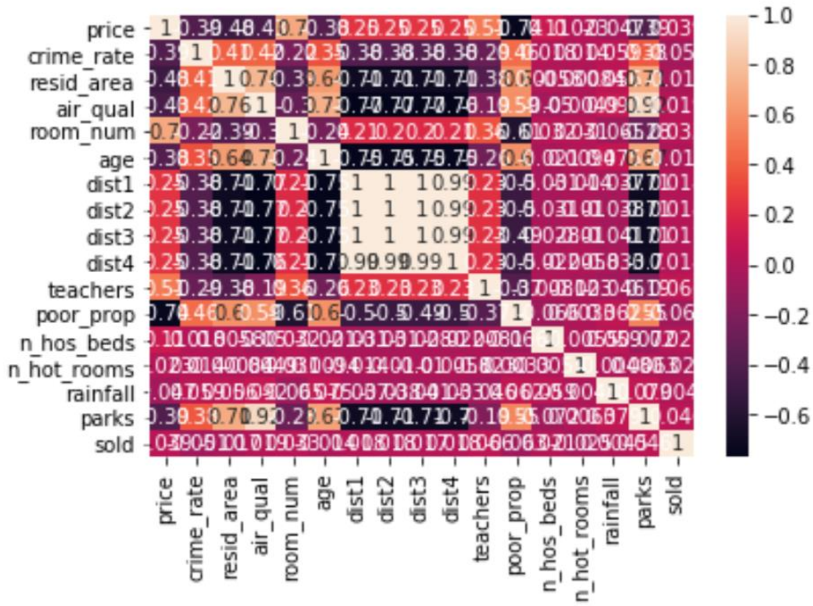
```
Out[16]: Text(27.125, 0.5, 'room_num')
```



SEABORN GRAPHS USING DATASET HOUSE PRICE

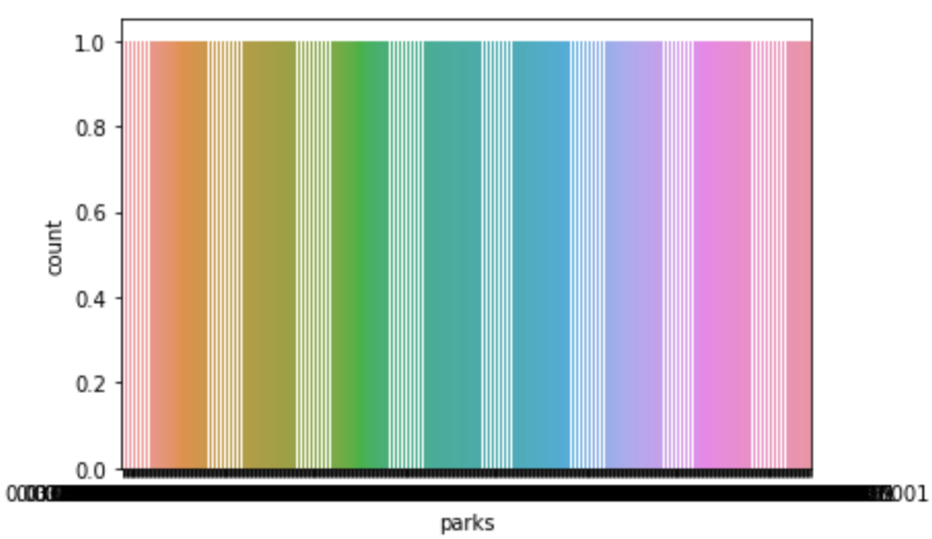
```
In [5]: sns.heatmap(df.corr(),annot=True)
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x13496889308>

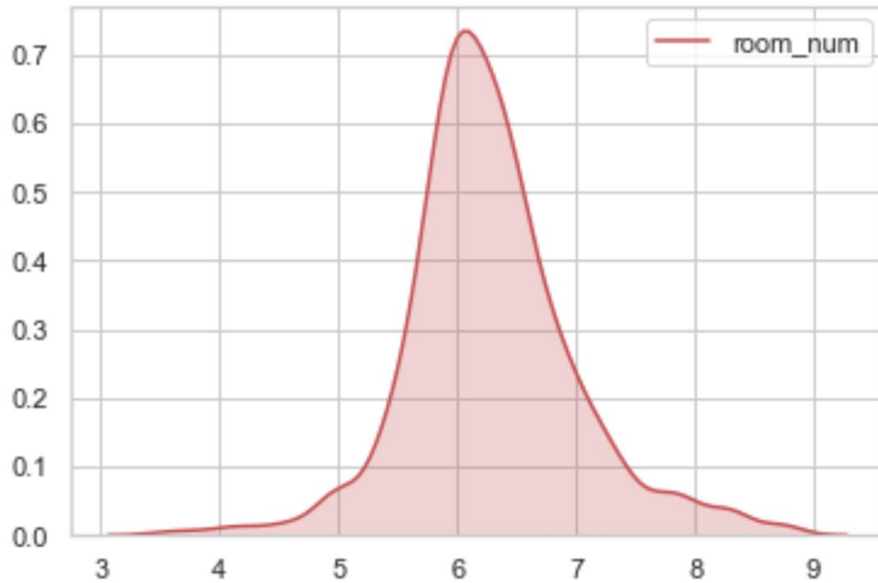



```
In [13]: sns.countplot(x="parks",data=df)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x18a4895a788>

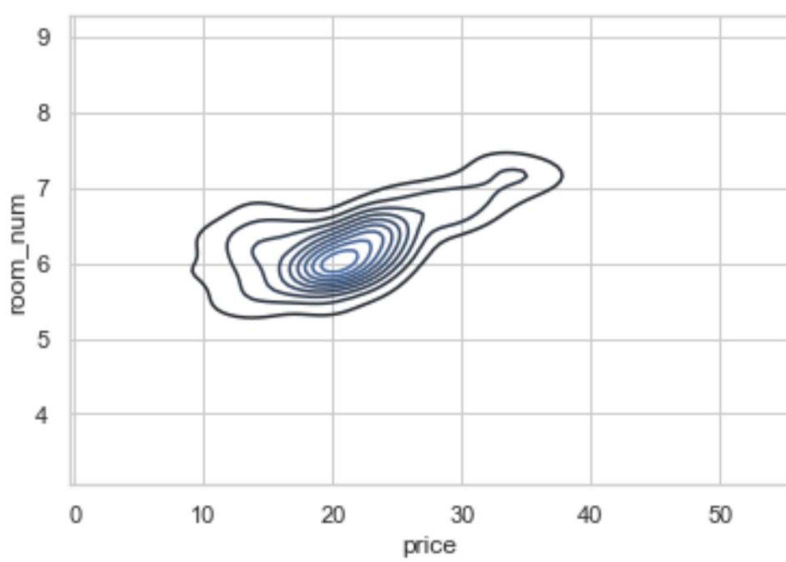


In [38]:
```
p1=sns.kdeplot(df["room_num"],shade=True,color="r")
```

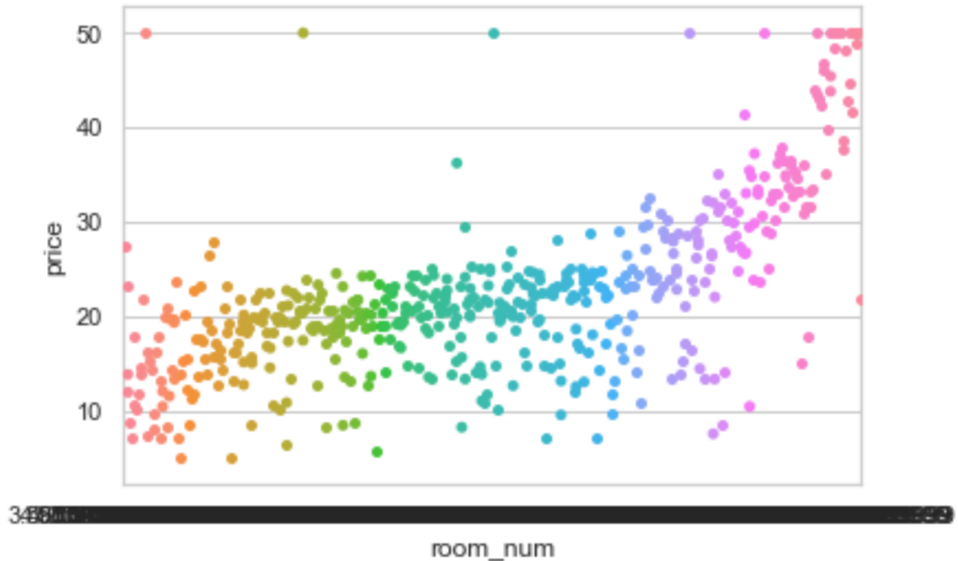


In [34]:
```
sns.kdeplot(df.price,df.room_num)
```

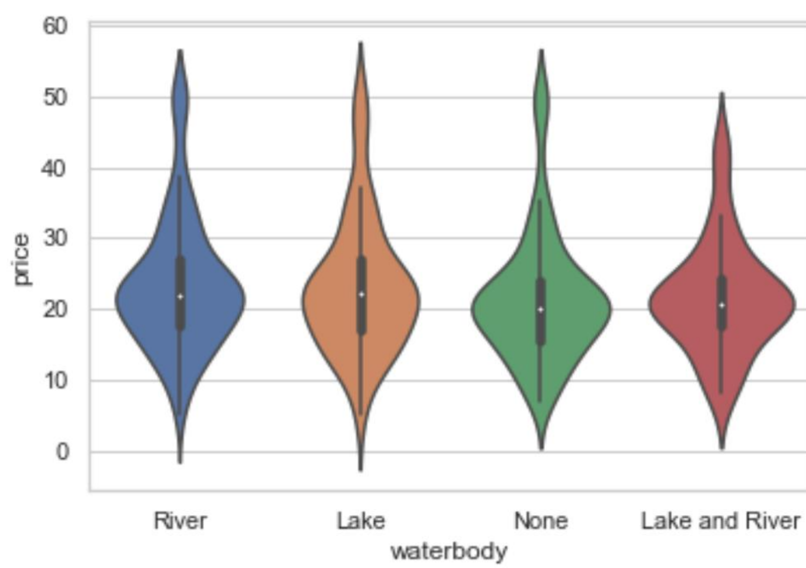
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1349e36fb08>



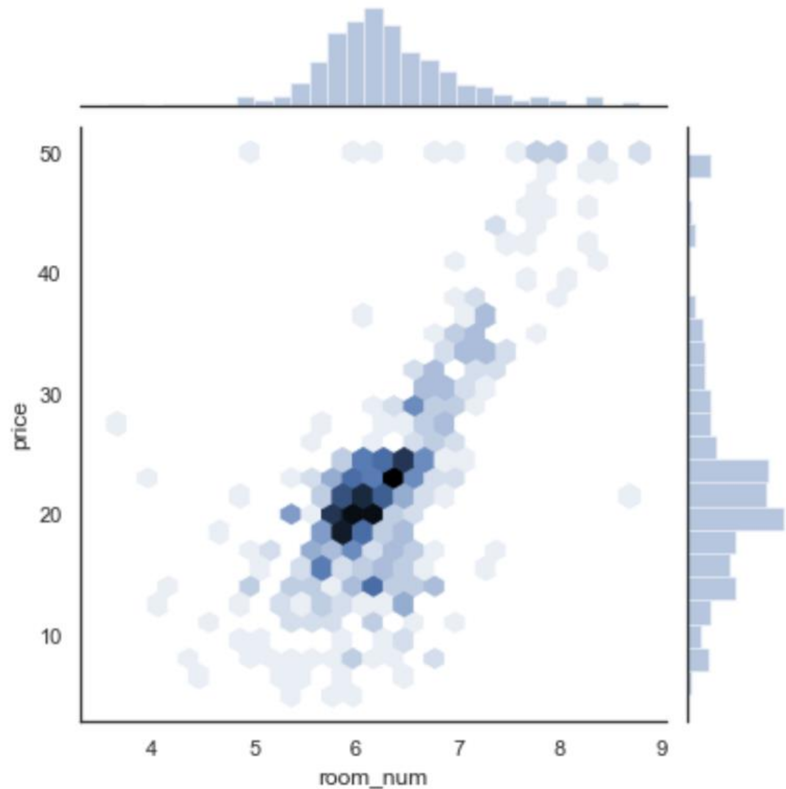
In [9]:
```
sns.stripplot(x="room_num",y="price",data=df)
```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1349a232848>



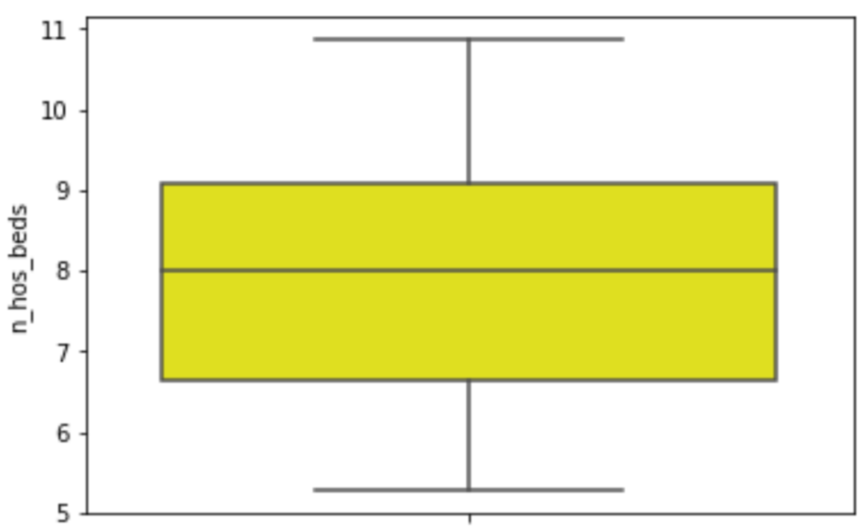
In [28]:
```
sns.violinplot(x="waterbody",y="price",data=df)
```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1349e1a8548>



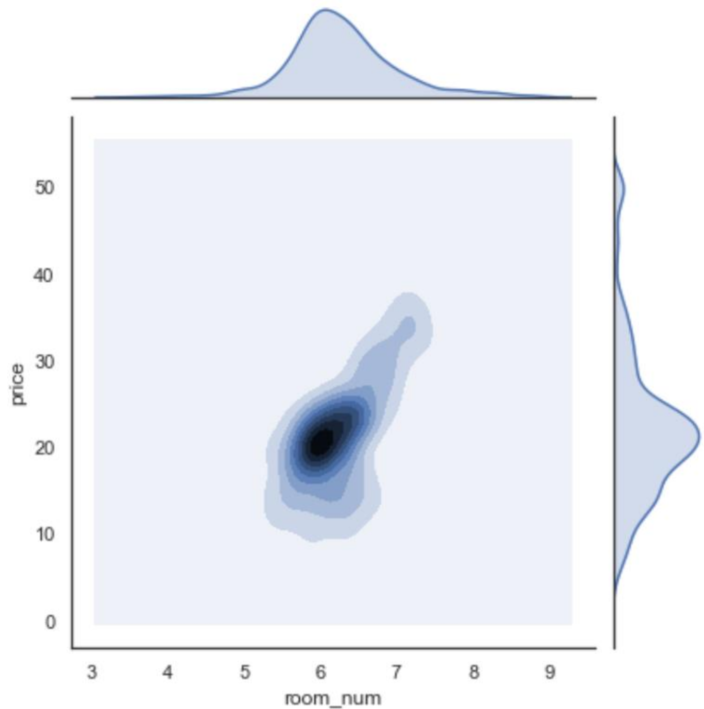
```
In [14]: with sns.axes_style('white'):
             sns.jointplot(x="room_num",y="price",data=df,kind='hex')
```



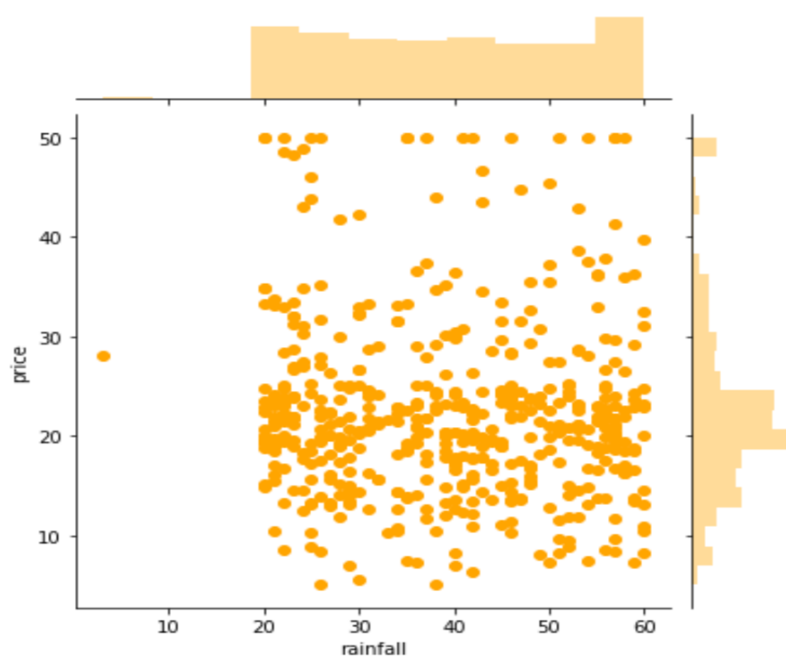

```
In [7]: sns.boxplot(y="n_hos_beds",data=df,color="yellow")
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x18dcd4af588>
```



```
In [13]: with sns.axes_style('white'):
             sns.jointplot(x="room_num",y="price",data=df,kind='kde')
```

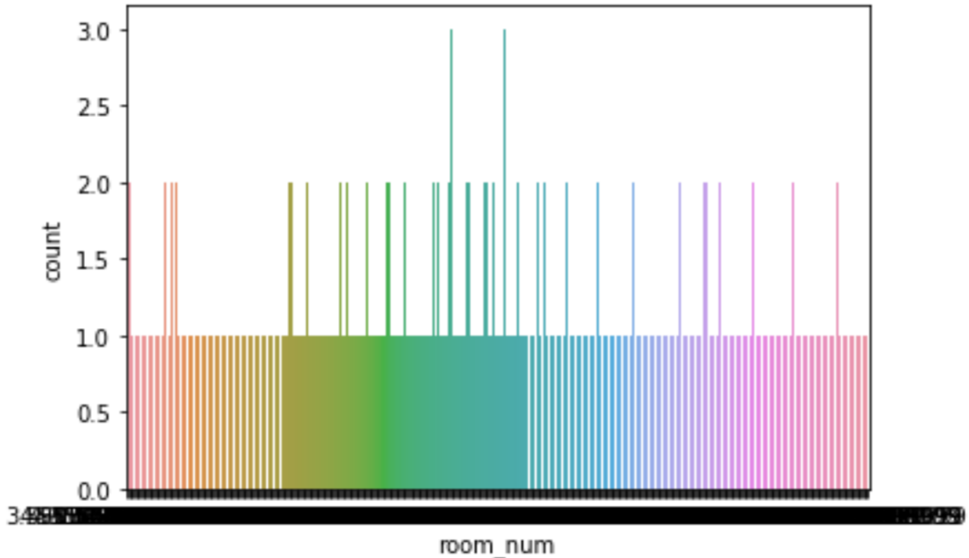



```
In [9]: sns.jointplot(x="rainfall",y="price",data=df,color="orange")
Out[9]: <seaborn.axisgrid.JointGrid at 0x18a485d3c88>
```



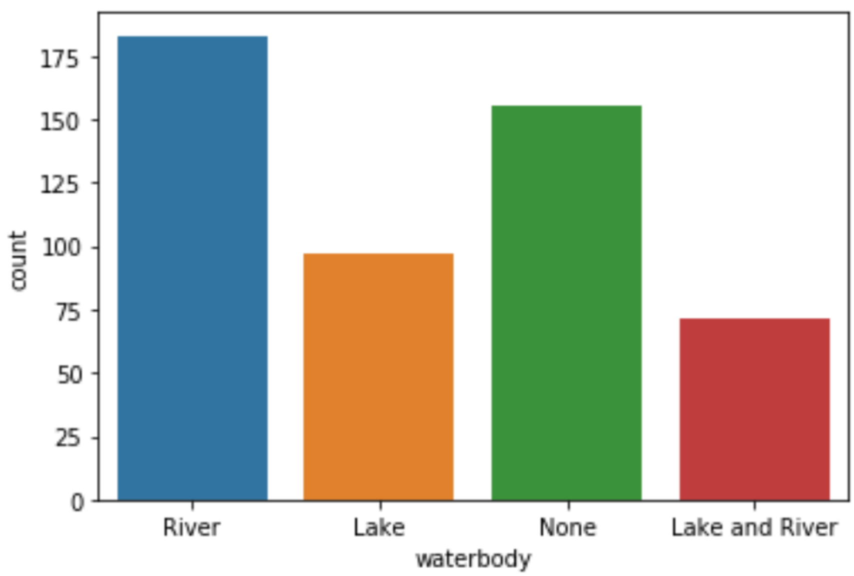
```
In [15]: sns.countplot(x="room_num",data=df)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x18a4a60c0c8>



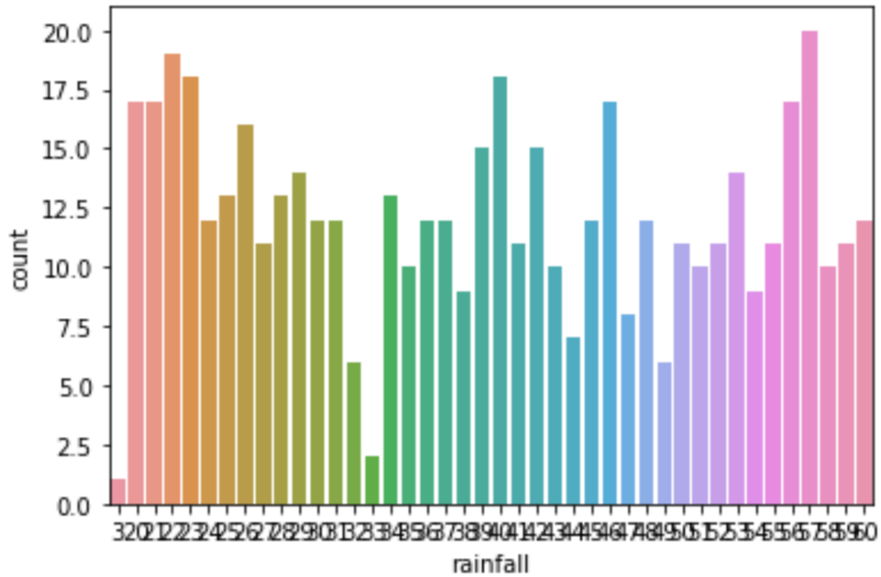

```
In [12]: sns.countplot(x="waterbody",data=df)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x18a48790248>



```
In [14]: sns.countplot(x="rainfall",data=df)
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x18a48695c88>
```



K-MEANS CLUSTERING

K-Means Clustering

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_blobs
```

```
dataset = make_blobs(n_samples=200,
                     centers=4,
                     n_features=2,
                     cluster_std=1.6,
                     random_state=50)
```

```
points = dataset[0]
```

```
# KMeans
from sklearn.cluster import KMeans
```

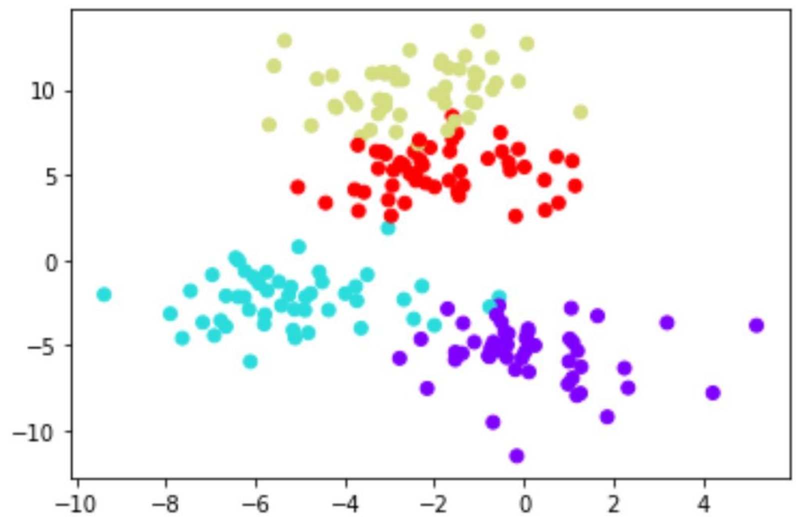
```
# Create a K means object
kmeans = KMeans(n_clusters=4)
```

```
# Fit this Kmeans object to this dataset
kmeans.fit(points)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```python
plt.scatter(dataset[0][:,0],dataset[0][:,1],c=dataset[1],cmap="rainbow")
```

```
<matplotlib.collections.PathCollection at 0x17dd07ff088>
```




```python
clusters = kmeans.cluster_centers_
```

```python
print(clusters)
```

```
[[-5.56465793 -2.34988939]
 [-2.40167949 10.17352695]
 [ 0.05161133 -5.35489826]
 [-1.92101646  5.21673484]]
```

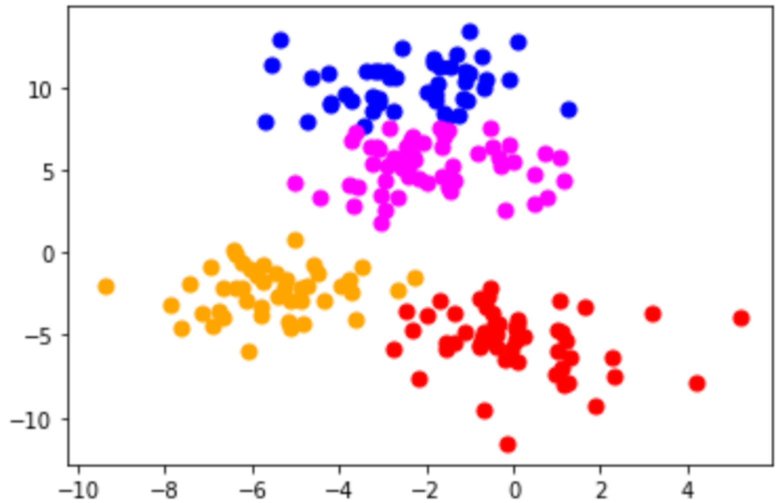
```python
y_km = kmeans.fit_predict(points)
```

```python
y_km
```

```
array([1, 0, 3, 1, 2, 0, 1, 0, 0, 2, 2, 1, 2, 3, 2, 3, 1, 1, 2, 2, 2, 3,
       1, 1, 1, 3, 1, 0, 3, 1, 1, 2, 2, 0, 0, 2, 2, 2, 3, 3, 1, 2, 3, 3,
       0, 0, 1, 3, 3, 1, 0, 2, 3, 1, 3, 3, 0, 2, 2, 3, 0, 0, 1, 0, 0, 1,
       0, 3, 1, 1, 0, 2, 1, 0, 3, 3, 2, 3, 3, 0, 2, 3, 2, 2, 2, 2, 1, 0,
       1, 2, 0, 1, 0, 3, 3, 2, 0, 3, 0, 3, 2, 0, 3, 1, 3, 0, 3, 0, 2, 1,
       0, 2, 1, 1, 2, 3, 1, 1, 3, 0, 1, 3, 1, 1, 0, 1, 1, 3, 3, 0, 2, 0,
       0, 3, 1, 3, 0, 1, 1, 2, 1, 3, 3, 0, 2, 1, 0, 3, 0, 2, 1, 2, 3, 0,
       0, 3, 1, 0, 0, 0, 3, 3, 0, 0, 2, 2, 2, 3, 0, 0, 0, 0, 2, 3, 0, 3,
       2, 0, 1, 3, 0, 3, 3, 1, 2, 0, 2, 1, 3, 0, 3, 3, 2, 3, 0, 1, 2, 3,
       2, 1])
```

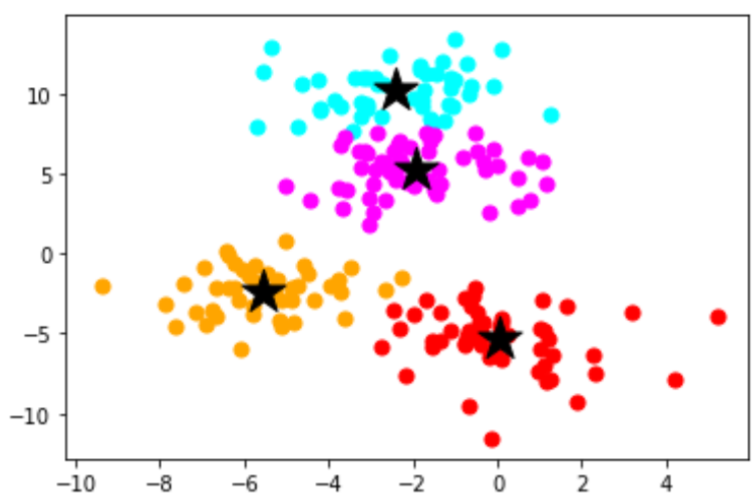
```python
plt.scatter(points[y_km == 0,0],points[y_km == 0,1],s=50,color="red")
plt.scatter(points[y_km == 1,0],points[y_km == 1,1],s=50,color="blue")
plt.scatter(points[y_km == 2,0],points[y_km == 2,1],s=50,color="orange")
plt.scatter(points[y_km == 3,0],points[y_km == 3,1],s=50,color="magenta")
```

```
<matplotlib.collections.PathCollection at 0x17dcf395e48>
```




```python
plt.scatter(points[y_km == 0,0],points[y_km == 0,1],s=50,color="red")
plt.scatter(points[y_km == 1,0],points[y_km == 1,1],s=50,color="cyan")
plt.scatter(points[y_km == 2,0],points[y_km == 2,1],s=50,color="orange")
plt.scatter(points[y_km == 3,0],points[y_km == 3,1],s=50,color="magenta")
plt.scatter(clusters[0][0],clusters[0][1],marker="*",s=500,color="black")
plt.scatter(clusters[1][0],clusters[1][1],marker="*",s=500,color="black")
plt.scatter(clusters[2][0],clusters[2][1],marker="*",s=500,color="black")
plt.scatter(clusters[3][0],clusters[3][1],marker="*",s=500,color="black")
```

```
<matplotlib.collections.PathCollection at 0x17dd0903348>
```

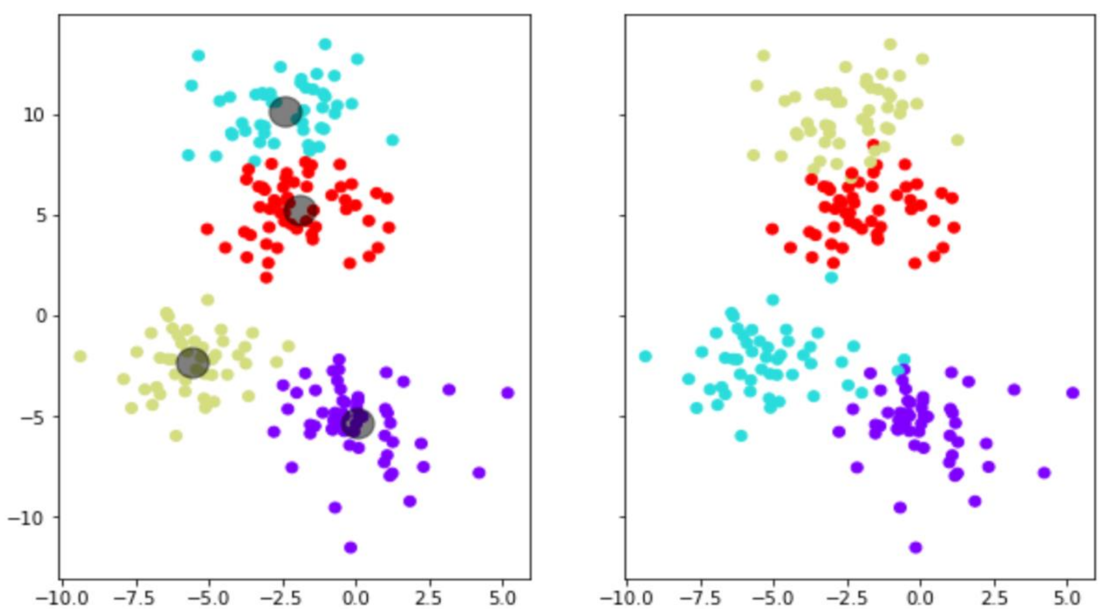


```
f, (ax1, ax2) = plt.subplots(nrows=1,
 ncols=2,
 sharey=True,
 figsize=(10,6))

ax1.scatter(dataset[0][:,0],
 dataset[0][:,1],
 c=y_km,
 cmap='rainbow')

ax2.scatter(dataset[0][:,0],
 dataset[0][:,1],
 c=dataset[1],
 cmap='rainbow')

ax1.scatter(x=clusters[:, 0],
 y=clusters[:, 1],
 c='black',
 s=300,
 alpha=0.5);
```



ADVANTAGES

The Python Package Index (PyPI) contains numerous Third-Party Modules that make python capable of interacting with most of the other languages and platforms.

Python language is developed under an OSI-approved open source license, which makes it free to use and distribute.

Python offers excellent readability and simple-to-learn syntax.

Python has in-built list and dictionary data structures which can be used to construct fast runtime data structures.

Python is ideal for general purpose tasks such as Machine Learning, Artificial Intelligence and Deep Learning.
Python provides a large standard library.

FUTURE SCOPE

- **The future scope of python is bright as it also helps in the analysis of large amount of data through its high-performance libraries and tools. The most popular Python libraries for the data visualization are MATPLOTLIB and SEABORN.**
- **In the field of Artificial Intelligence, Python is used as an engineering tool. The scope of Artificial Intelligence with python is pretty wide and being open source people will contribute to it and keep it going.**
- **Python has numerous of frameworks, libraries like Sk-learn, Numpy, Pandas, Seaborn, Matplotlib, and many more which has made python so popular.**
- **The future scope of Python deals with analyzing a large number of data sets across computer clusters through its high-performance toolkits and libraries.**

CONCLUSION

There are no doubts that AI technologies are the future. Considering the increasing popularity of the trend and the number of people ready to invest in it, the global AI market is going to reach $89.8 billion by 2025. The PL is what we should think about at first. The complexities of coding as well as the availability of the experienced and qualified developers are crucial moments to take into account as well. We're to deal and process a host of data effectively when it comes to AI industry The marketing can make use of AI by means of the tech stack of the processes that are made manually by employees can be automated, it can bring more efficiency and quickly analyze large data sets, for example. Gartner says that by 2020 AI technologies will be used in at least one of the sales processes by 30% of companies over the world. Besides that, according to Accenture reports, the profitability willrise by 38% by 2035 and AI will create $14 trillion of additional revenue. The e-commerce sales are expected to be about $4.5 trillion by 2021. And that's not without AI technologies used. Thanks to the AI the sites provides the customers with 24/7 service and assistance by means of the chatbots, improve consumers

experience by analyzing the CRM data in moments with AI tech, IoT, and other examples of using AI in e-commerce. High diversity of built-in libraries, simple syntax, readability, compatibility, rapid testing of sophisticated algorithms, accessibility to non-programmers, and other features make Python worthy of your attention. All that ease the process, save your budget and increase the popularity of Python. Taking to account all the advantages we get using the PL, the conclusion is obvious — Python is what we need to consider to your AI-based project.

BIBLIOGRAPHY

The contents have been gathered from the following:

- **Information: GOOGLE**
- **Images: GOOGLE IMAGES**
- **Snapshots: Self-performed**