PROJECT ON

DATA SCIENCE WITH PYTHON

SUMBMITTED BY=Adhyayan Srijan

SUBMITTED TO=ANKIT PRAMANIK

SUBMITTED ON=16/07/2019

Email.id=adhyayansrijan1998@gmail.com

Phone no.=8709394277

CONTENTS

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. ANKIT PRAMANIK and Mr. SAYANTAN CHAKRABORTY for their support, co-operation and encouragement which helped me in completion of this project.

INTRODUCTION

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. The list keeps going and never ends. Python has an easy implementation for OpenCV. What makes Python favourite for everyone is its powerful and easy implementation. For other languages, students and researchers need to get to know the language before getting into ML or AI with that language. This is not the case with python. Even a programmer with very basic knowledge can easily handle python. Apart from that, the time someone spends on writing and debugging code in python is way less when compared to C, C++ or Java. This is exactly what the students of AI and ML want. They don't want to spend time on debugging the code for syntax errors, they want to spend more time on their algorithms and heuristics related to AI and ML. Not just the libraries but their tutorials, handling of interfaces are easily available online. People build their own libraries and upload them on GitHub or elsewhere to be used by others.

OBJECTIVES

- Python is a powerful open source programming language, which means that it's free to use while having all the properties that a programming language should have. • It is a versatile programming language that supports ObjectOriented Programming, Structured Programming, and functional programming patterns.
- Python has some 72,000 libraries in the Python Package Index that aid in scientific calculations and machine learning applications.
- Python sports an easy to understand and readable syntax that ensures that the development time is cut into half when compared with other programming languages.
- Python enables you to perform data analysis, data manipulation, and data visualization, which are very important in data science.

HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements

Operating System : Windows/Linux
Front End : Python 3.7

Platform : Anaconda

Hardware requirements

Speed : 233MHz and above
Hard disk : 10GB
RAM : 256 MB

FUTURE SCOPE

➢ Python programming language is undoubtedly dominating the other languages when future technologies like Artificial Intelligence(AI) comes into the play.

➢ There are plenty of python frameworks, libraries, and tools that are specifically developed to direct Artificial Intelligence to reduce human efforts with increased accuracy and efficiency for various development purposes. ➢ It is only the Artificial Intelligence that has made it possible to develop speech recognition system, autonomous cars, interpreting data like images, videos etc.

ADVANTAGES

1) Python can be used in the development of prototypes, and it can help speed up the concept to creation process because it is so easy to use and read.

2) Python is ideal for general purpose tasks such as data mining, and big data facilitation

CODING

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
%matplotlib inline
```

In [2]:

```
np.zeros(10) #used to display an array of n zeros in float data type where n is a posit
ive, real,whole no.
             #given inside the paranthesis
```

Out[2]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [3]: np.zeros(10,int) #if we want integer instead of float just use

```
np.zeros(10,int)
```

Out[3]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [4]:

```
np.ones(10)#used to display an array of n ones in float data type where n is a positiv
e, real,whole no.
             #given inside the paranthesis
```

Out[4]:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [5]: np.ones(10,int) #if we want integer instead of float just use

```
np.zeros(10,int)
```

Out[5]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [6]:

```
np.arange(10,51) #used to create an array of elements starting from the first no. given
inside the paranthesis to second
                #no. in the paranthesis please note that the second no. in the paranthe
sis is not included as an element of
                # the array
```

Out[6]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
 44, 45, 46, 47, 48, 49, 50])
```

In [7]:

```
np.arange(10,51,3) #the third no. makes the elements jump values in the order of the n
o. itself  in this case 3 as we all
                   #can see that after 10 the next element is 13 intead of 11
```

Out[7]:

```
array([10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49])
```

In [8]:

```
a=np.arange(0,25)
b=a.reshape(5,5) #the reshape method is used to covert a 1D array to a 2D array and the
rows and columns of the 2D matrix b                # are the no.s inside the paranthesis
```

Out[8]:

```
array([[ 0,  1,  2,  3,  4],
[ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],

[20, 21, 22, 23, 24]])
```

In [9]:

```
b.flatten() #used to convert a 2D matrix to 1D matrix
```

Out[9]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

In [10]:

```
np.eye(3) #used to create an identity matrix of dimensions mentioned inside the paranth
esis since it is going to be square
        # matrix i.e rows and columns both are equal so only one argument is passed des
cibing both the no. of rows and columns
```

Out[10]:

```
array([[1., 0., 0.],
[0., 1., 0.],        [0.,
0., 1.]])
```

In [11]:

```
np.random.rand() #used to give any random number of float
datatype
```

Out[11]:

```
0.46009273293455133
```

In [12]:

```
np.random.rand(10) #used to give an array of n random elements of float datatype where
 n is the no. inside the paranthesis
```

Out[12]:

```
array([0.24740306, 0.10204996, 0.9096343 , 0.51445138, 0.0479171 ,
0.53495508, 0.72004934, 0.34589298, 0.98337697, 0.97069257])
```

In [13]:

```
np.random.randint(1,10) #used to give any random no. in the range of the no.s given ins
ide the paranthesis Out[13]:
```

6

In [14]:

```
np.random.randint(1,10,5) #used to give an array of n random elements in the range of t
he first two no.s given inside
                          #the paranthesis where n is the third no. given in the parant
hesis Out[14]:
```

```
array([7, 2, 3, 1, 3])
```

In [15]:

```
np.linspace(1,10,9).reshape(3,3) #linspace is used to create an array of elements which
are formed by n equal division of the
                                  #the range created by the first two no.s in the paran
thesis where n is the third no.
                                #and as mentioned above the reshape method is used to c
overt a 1D array to a 2D array. Out[15]:
```

```
array([[ 1.   ,  2.125,  3.25 ],
[ 4.375,  5.5  ,  6.625],
       [ 7.75 ,  8.875, 10.   ]])
```

In [16]:

```
mat=np.arange(1,26)
mat
```

Out[16]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
 18, 19, 20, 21, 22, 23, 24, 25])
```

In [17]:

```
mat[1] #used to give the no. at any index where the index is specified by the no. given
inside the square brackets Out[17]:
```

2

In [18]:

```
mat[2:6] #used to give an array of the no.s present in between the index values specifi
ed in the square bracket
         #the index value of the first no. is included whearas of the second no. is neg
lected
```

Out[18]:

```
array([3, 4, 5, 6])
```

In [19]:

```
mat[:5] #if nothing is given then it means that array will start with the element prese
nt in the first index value
         #if end is not specified array will continue to the element present at the las
t index value 
```

Out[19]:

```
array([1, 2, 3, 4, 5])
```

In [20]:

```
mat1=mat.reshape(5,5)
mat1
```

Out[20]:

```
array([[ 1,  2,  3,  4,  5],
[ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
[21, 22, 23, 24, 25]])
```

In [21]:

```
mat1[2] #gives an array containing elements present in the row of nth index value where
n is the no. in the square bracket 
```

Out[21]:

```
array([11, 12, 13, 14, 15])
```

In [22]:

```
mat1[3][4] #gives the element present in the row of nth index value and column of mth i
ndex value where n is the no. in the
          # first square bracket and m is the no. in the second square bracket
```

Out[22]:

```
20
```

In [23]:

```
mat1[2:,:4] #used to give an array that contains elements from and till a given index v
alue
            #[n:,:m] if ':' is given after the no. that means onwards it(rows and colum
ns as per the use) (including it)
            #and if ':' is used it means till it (rows and columns as per the use) (not
including it) 
```

Out[23]:

```
array([[11, 12, 13, 14],
[16, 17, 18, 19],
       [21, 22, 23, 24]])
```

In [24]:

```
mat1[2:4,3:4] #using the above concept we can apply it to be used in more complex situa
tions Out[24]:
```

array([[14],

[19]]) In [25]:

```
np.sum(mat1) #sum of each elements in the matrix
```

Out[25]:

325

In [26]: np.sum(mat1,axis=0) #array of sum of elements

```
column wise
```

Out[26]:

array([55, 60, 65, 70, 75])

In [27]:

```
np.sum(mat1,axis=1) #array of sum of elements row wise
```

Out[27]:

array([15, 40, 65, 90, 115])

In [28]:


```
a=np.arange(1,10)
```

In [29]:

```
a+a
```

Out[29]:

```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18])
```

In [30]:

```
a-a
```

Out[30]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [31]:

```
a*a
```

Out[31]:

```
array([ 1,  4,  9, 16, 25, 36, 49, 64, 81])
```

In [32]:

```
a/a
```

Out[32]:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [33]:

```
np.sqrt(a) #give an array containing the elements which are square root of all elements in the array
```

Out[33]:

```
array([1.        , 1.41421356, 1.73205081, 2.        , 2.23606798,
       2.44948974, 2.64575131, 2.82842712, 3.        ])
```

In [34]:

```
np.log(a) #give an array containing the elements which are log of all elements in the array
```

Out[34]:

```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791,
       1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

In [35]:

```
np.std(a) #give an array containing the elements which are standard deviation of all elements in the array
```

Out[35]:

```
2.581988897471611
```

In [36]:

```
sal=pd.read_csv('Salaries.csv') #used to read the csv file in the jupyter notebook be s
ure that the csv file is in same
                                #folder as the jupyter notebook
```

In [37]:

```
sal.head() #to display the first few rows of the dataframe,ww can also choose the no. o
f rows that we have to display just by
           #specifying so in the paranthesis as shown in the next cell
           #ex- usa.head(10) will show first 10 rows simiarly usa.head(n) will show n r
ows where n is a positive whole number.
```

Out[37]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalP
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373

In [38]:

```
sal.head(10)
```

Out[38]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalP
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343

4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373
5	6	DAVID SULLIVAN	ASSISTANT DEPUTY CHIEF II	118602.00	8601.00	189082.74	NaN	316285
6	7	ALSON LEE	BATTALION CHIEF, (FIRE DEPARTMENT)	92492.01	89062.90	134426.14	NaN	315981
7	8	DAVID KUSHNER	DEPUTY DIRECTOR OF INVESTMENTS	256576.96	0.00	51322.50	NaN	307899
8	9	MICHAEL MORRIS	BATTALION CHIEF, (FIRE DEPARTMENT)	176932.64	86362.68	40132.23	NaN	303427
9	10	JOANNE HAYES-WHITE	CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	285262.00	0.00	17115.73	NaN	302377

In [39]:

```
sal.info() #to display the information related to the no. of rows and columns of the da
taframe
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
148654 entries, 0 to 148653
Data columns (total 13 columns):
Id                  148654 non-null int64
EmployeeName        148654 non-null object
JobTitle            148654 non-null object
BasePay             148045 non-null float64
OvertimePay         148650 non-null float64
OtherPay            148650 non-null float64
Benefits            112491 non-null float64
TotalPay            148654 non-null float64
TotalPayBenefits    148654 non-null float64
Year                148654 non-null int64
Notes               0 non-null float64
Agency              148654 non-null object
Status              0 non-null float64 dtypes:
float64(8), int64(2), object(3) memory usage:
14.7+ MB
```

In [40]:

```
sal.describe() #used to display the maximum,minimum,mean,standard deviation of the data
frame and aslo other statistical details Out[40]:
```

	Id	BasePay	OvertimePay	OtherPay	Benefits	Tot
count	148654.000000	148045.000000	148650.000000	148650.000000	112491.000000	148654.00

mean	74327.500000	66325.448841	5066.059886	3648.767297	25007.893151	74768.32
std	42912.857795	42764.635495	11454.380559	8056.601866	15402.215858	50517.00
min	1.000000	-166.010000	-0.010000	-7058.590000	-33.890000	-618.13
25%	37164.250000	33588.200000	0.000000	0.000000	11535.395000	36168.99
50%	74327.500000	65007.450000	0.000000	811.270000	28628.620000	71426.6
75%	111490.750000	94691.050000	4658.175000	4236.065000	35566.855000	105839.13
max	148654.000000	319275.010000	245131.880000	400184.250000	96570.660000	567595.43

In [41]:

```
sal['BasePay'] #give all the rows of the given column
```

Out[41]:

```
0         167411.18
1         155966.02
2         212739.13
3         77916.00
4         134401.60
5         118602.00
6         92492.01
7         256576.96
8         176932.64
9         285262.00
10        194999.39
11        99722.00
12        294580.02
13        271329.03
14        174872.64
15        198778.01
16        268604.57
17        140546.87
18        168692.63
19        257510.59
20        257510.48
21        257510.44
22        140546.88
23        168692.63
24        140546.86
25        256470.41
26        92080.80
27        168692.59
28        261717.60
29        246225.60               ...
148624         0.00
148625         0.00
148626         0.00
148627         0.00
148628         0.00
148629         0.00
148630         0.00
148631         0.00
148632         0.00
148633         0.00
148634         0.00
148635         0.00
148636         0.00
148637         0.00
148638         0.00
148639         0.00
148640         0.00
148641         0.00
148642         0.00
148643         0.00
148644         0.00
148645         0.00
148646         NaN
148647         0.00
148648         0.00
```

```
148649          0.00
148650          NaN
148651          NaN
```

```
148652          NaN
148653          0.00
Name: BasePay, Length: 148654, dtype: float64
```

In [42]:
```
sal['BasePay'].mean() #mean values of all the rows of the given column
```

Out[42]:

```
66325.44884050643
```

In [43]:
```
sal['BasePay'].max() #max values of all the rows of the given column
```

Out[43]:

```
319275.01
```

In [44]:

```
a=sal[sal['EmployeeName']=='JOSEPH DRISCOLL']
a
```

Out[44]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalP
24	25	JOSEPH DRISCOLL	CAPTAIN, FIRE SUPPRESSION	140546.86	97868.77	31909.28	NaN	270324

In [45]:
```
a=sal[sal['TotalPay']==sal['TotalPay'].max()]
a['EmployeeName']
```

Out[45]:

```
0    NATHANIEL FORD
Name: EmployeeName, dtype: object
```

In [46]:

```
a=sal[sal['TotalPay']==sal['TotalPay'].min()]
print(a)
```

```
            Id EmployeeName                   JobTitle  BasePay  Overtime
Pay  \
148653  148654    Joe Lopez  Counselor, Log Cabin Ranch      0.0
0.0

        OtherPay  Benefits  TotalPay  TotalPayBenefits  Year  Notes  \
148653   -618.13       0.0   -618.13           -618.13  2014    NaN

               Agency  Status
```

```
148653   San Francisco     NaN
```

In [47]:

```
a=sal.loc[sal['TotalPay'].idxmax()]
a
```

Out[47]:

```
Id                                                               1
EmployeeName                                        NATHANIEL FORD
JobTitle            GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY
BasePay                                                     167411
OvertimePay                                                      0
OtherPay                                                    400184
Benefits                                                       NaN
TotalPay                                                    567595
TotalPayBenefits                                            567595
Year                                                          2011
Notes                                                          NaN
Agency                                               San Francisco
Status                                                         NaN
Name: 0, dtype: object
```

In [48]:

```
a=sal.groupby('Year').mean()['TotalPay']
a
```

Out[48]:

```
Year
2011    71744.103871
2012    74113.262265
2013    77611.443142
2014    75463.918140 Name: TotalPay, dtype: float64 In [49]:
```

```
sal['JobTitle'].nunique()
```

Out[49]:

2159 In

[50]:

```
a=sal['BasePay'].value_counts()
a.head(10)
```

Out[50]:

```
0.0         1298 54703.0
338
55026.0      297
48472.4      210
65448.0      153
68391.0      152
121068.0     152
88374.0      151
51492.8      143
94191.0      137
```

Name: BasePay, dtype: int64

In [51]:

```
sum(sal['JobTitle'].value_counts()==1)
```

Out[51]:

239 In

[52]:

```
a=sal[sal['Year']==2013]
sum(a['JobTitle'].value_counts()==1)
```

Out[52]:

202 In

[53]:

```
def a (x):
    if 'Chief' in x:
        return True
    else:
        return False
a=sal['JobTitle'].apply(lambda x:a(x))
sum(a)
```

Out[53]:

423 In

[54]:

```
sal['b']=sal['JobTitle'].apply(len)
a=sal[['b','TotalPayBenefits']]
a.corr()
```

Out[54]:

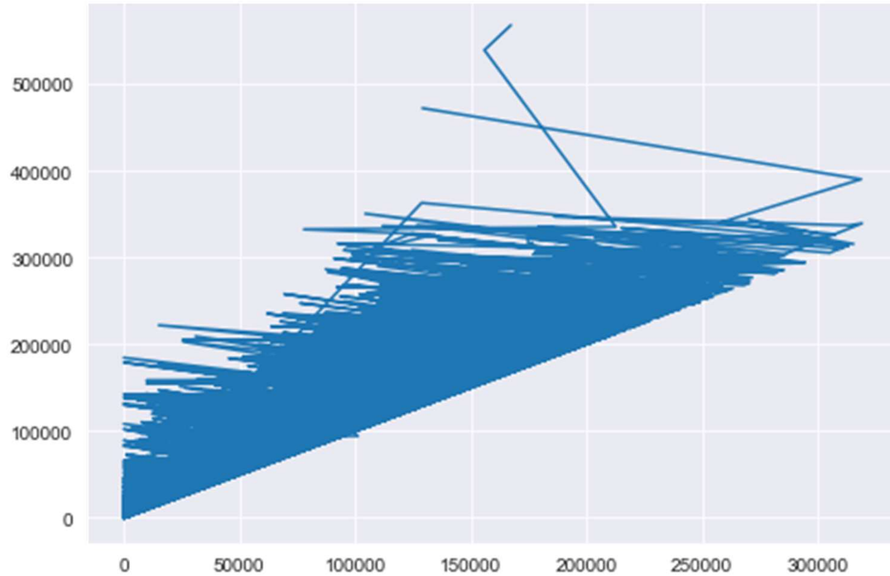
	b	TotalPayBenefits
b	1.000000	-0.036878
TotalPayBenefits	-0.036878	1.000000

In [55]:

```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(sal['BasePay'],sal['TotalPay'])
```

Out[55]:

```
[<matplotlib.lines.Line2D at 0x1e6e8f08208>]
```

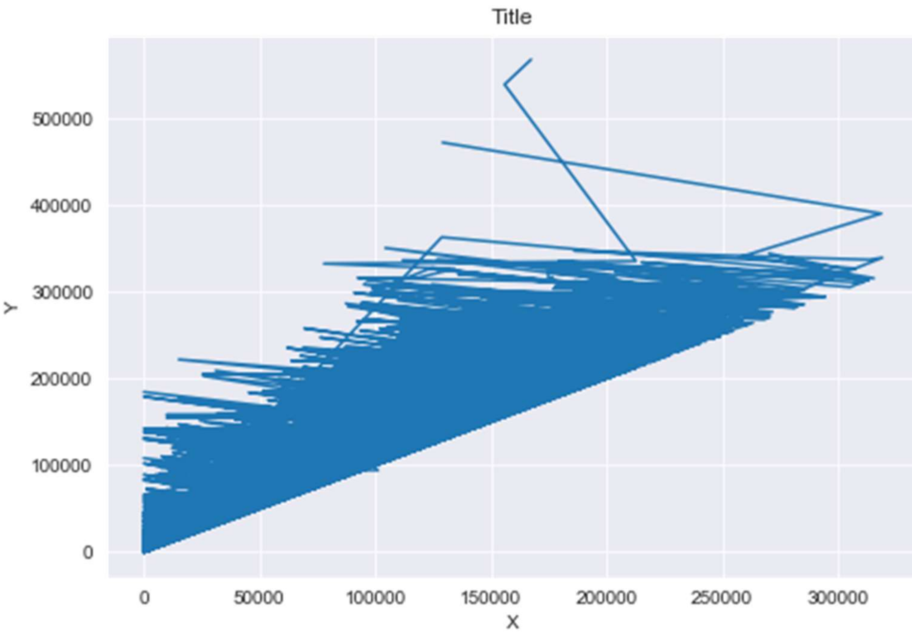

500000
400000
300000
200000
100000
0
0
50000
100000
150000
200000
250000
300000

In [56]:

```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(sal['BasePay'],sal['TotalPay'])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_title('Title')
```

Out[56]:

```
Text(0.5, 1.0, 'Title')
```



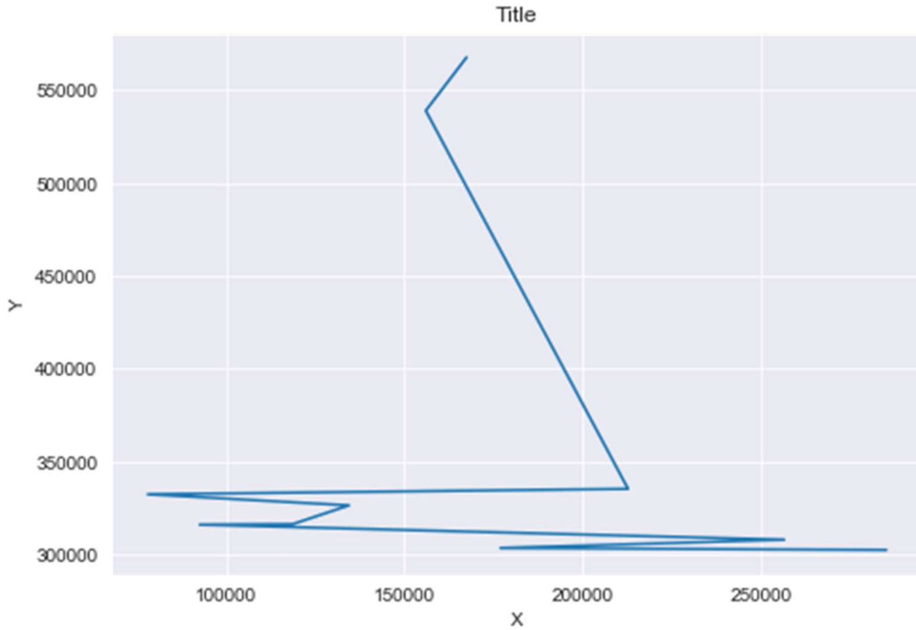
In [57]:

```
#since,graph is not clear we can take only first 10 values also.

fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
w=sal['BasePay'].head(10)
e=sal['TotalPay'].head(10)
ax.plot(w,e)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_title('Title')
```

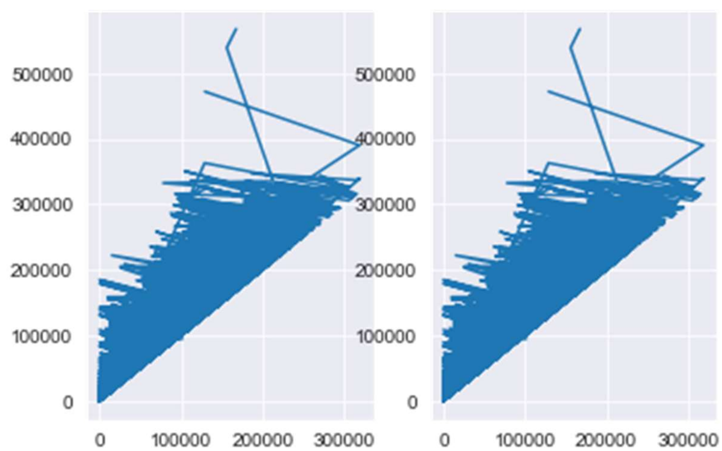
Out[57]:

```
Text(0.5, 1.0, 'Title')
```



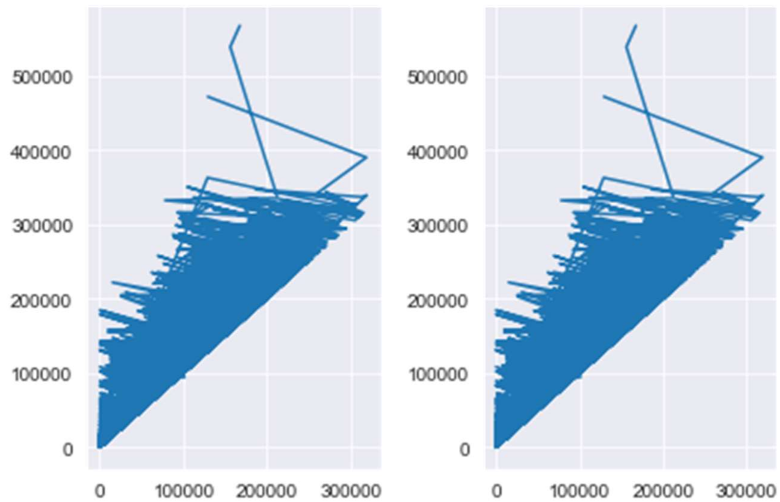
In [58]:

```
fig,ax=plt.subplots(nrows=1,ncols=2)
for i in ax:
    i.plot(sal['BasePay'],sal['TotalPay'])
```



In [59]:

```
fig,ax=plt.subplots(nrows=1,ncols=2)
for i in ax:
    i.plot(sal['BasePay'],sal['TotalPay'])
plt.tight_layout()
```

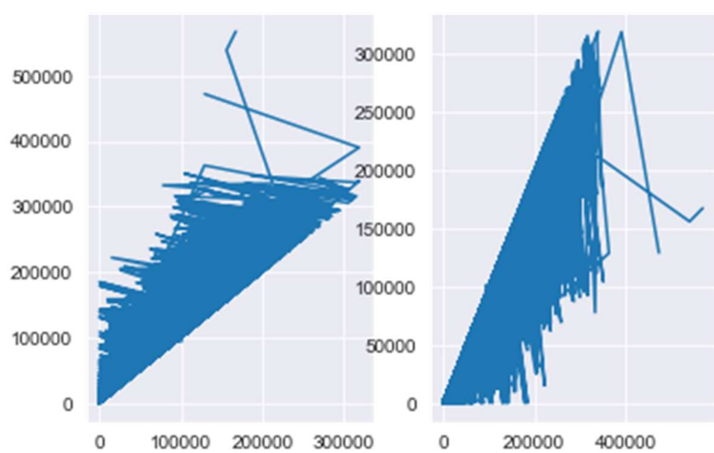


In [60]:

```
fig,ax=plt.subplots(nrows=1,ncols=2)
ax[0].plot(sal['BasePay'],sal['TotalPay'])
ax[1].plot(sal['TotalPay'],sal['BasePay'])
```

Out[60]:

```
[<matplotlib.lines.Line2D at 0x1e6f3e9be10>]
```

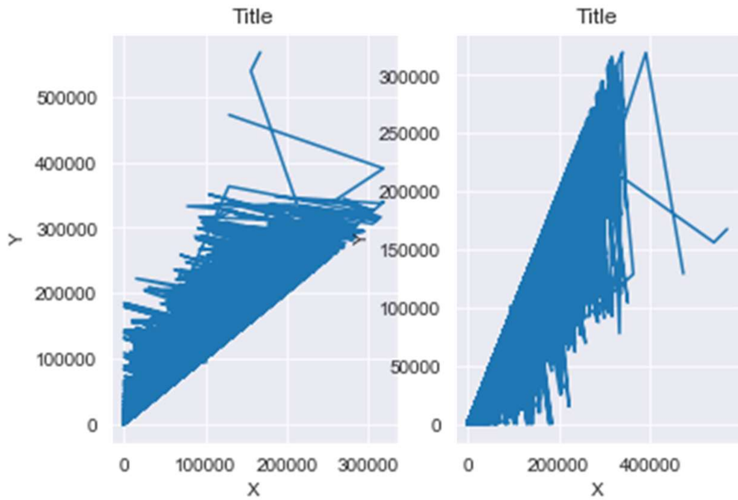


In [61]:

```
fig,ax=plt.subplots(nrows=1,ncols=2)
ax[0].plot(sal['BasePay'],sal['TotalPay'])
ax[0].set_xlabel('X')
ax[0].set_ylabel('Y')
ax[0].set_title('Title')
ax[1].plot(sal['TotalPay'],sal['BasePay'])
ax[1].set_xlabel('X')
ax[1].set_ylabel('Y')
ax[1].set_title('Title')
```

Out[61]:

```
Text(0.5, 1.0, 'Title')
```

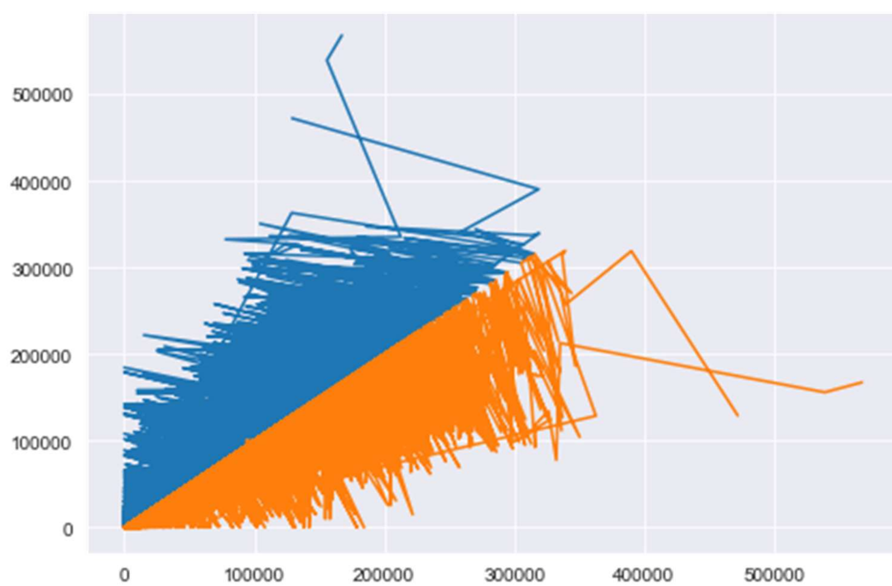


In [62]:

```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(sal['BasePay'],sal['TotalPay'])
ax.plot(sal['TotalPay'],sal['BasePay'])
```

Out[62]:

```
[<matplotlib.lines.Line2D at 0x1e6800124e0>]
```

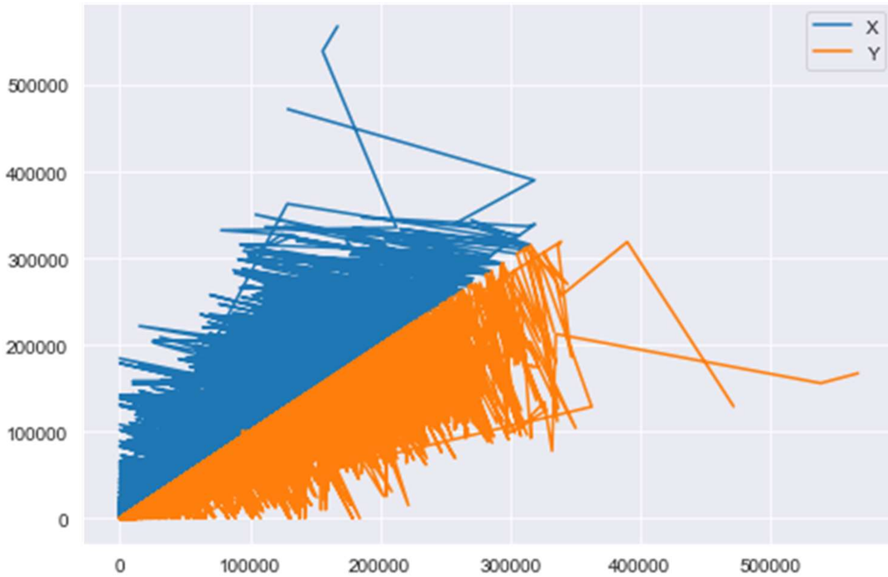


In [63]:

```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(sal['BasePay'],sal['TotalPay'],label='X')
ax.plot(sal['TotalPay'],sal['BasePay'],label='Y')
ax.legend(loc=0)
```

Out[63]:

```
<matplotlib.legend.Legend at 0x1e6e976ce48>
```

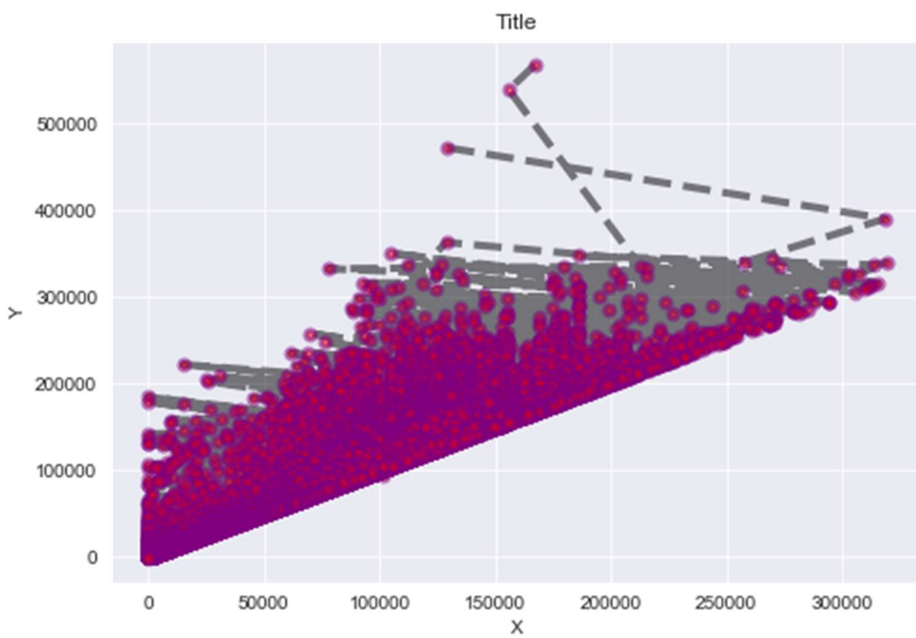


In [64]:

```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.plot(sal['BasePay'],sal['TotalPay'],color='black',lw=4,ls='--',alpha=0.5,marker='o',
markersize=5,markerfacecolor='red',markeredgewidth=3,markeredgecolor='purple')
ax.set_xlabel('X') ax.set_ylabel('Y') ax.set_title('Title') #color=color of the lines
#lw=linewidth=thickness of the lie=nes
#ls=linestyle=style of lines
#alpha=transperancy of the lines
#marker=type of marker used to plot the points on the graph
#markersize=size of marker
#markeredgewidth=thickness of the edge of the marker
#markeredgecolor=color of marker's edge
#
```

Out[64]:

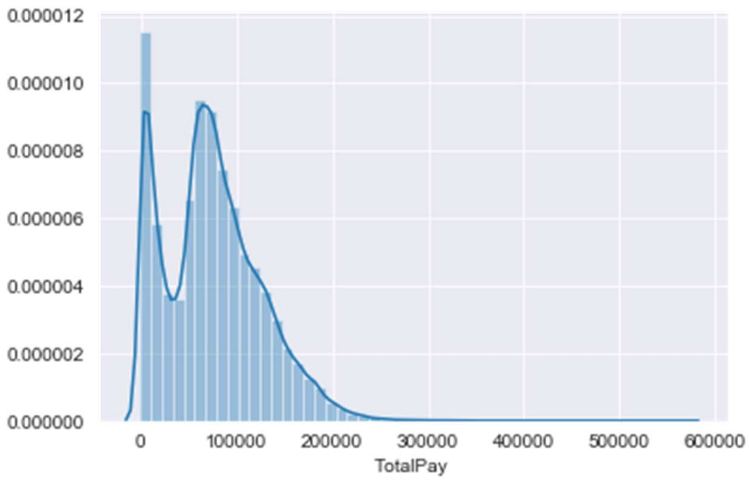
```
Text(0.5, 1.0, 'Title')
```



In [65]:

```
sns.distplot(sal['TotalPay'])
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x1e682fa0c18>

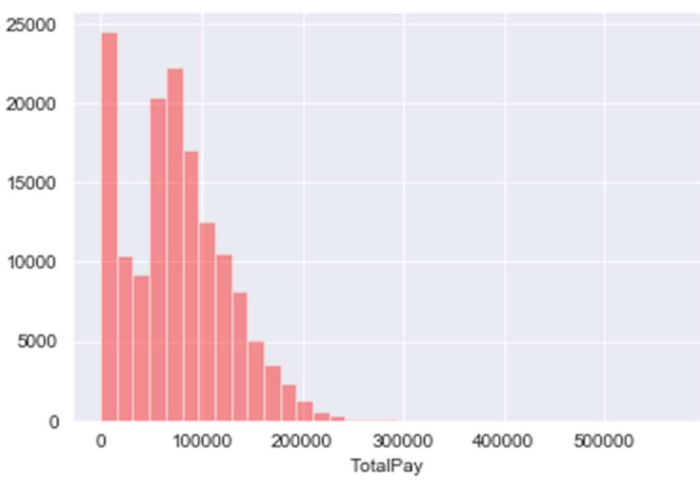


In [66]:

```
sns.distplot(sal['TotalPay'],color='red',bins=35,kde=False)
```

Out[66]:

<matplotlib.axes._subplots.AxesSubplot at 0x1e682fa0320>



In [67]:

```
sns.jointplot(x='TotalPayBenefits',y='BasePay',data=sal,kind='scatter')
```

Out[67]:

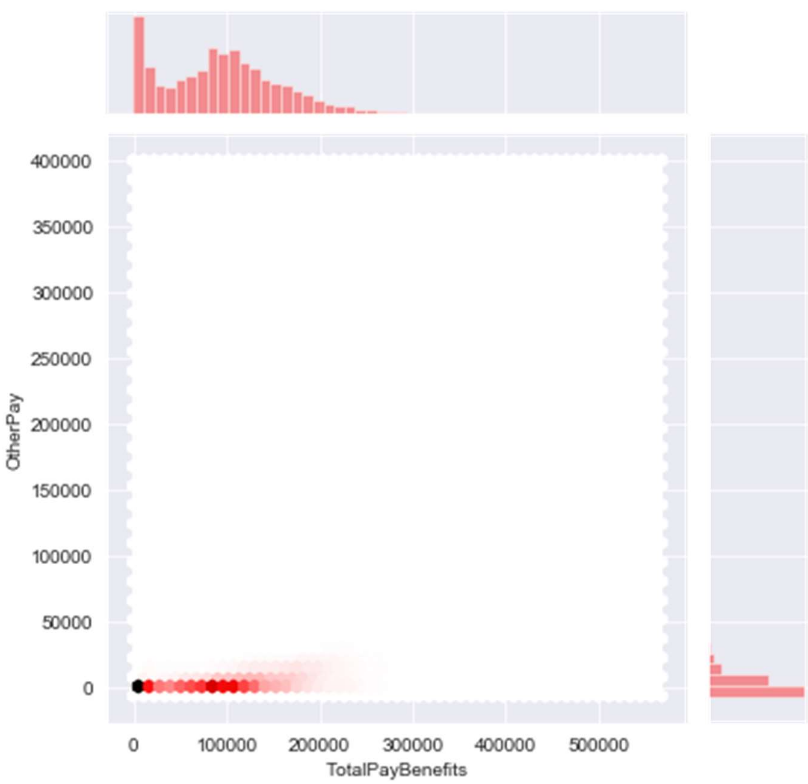
<seaborn.axisgrid.JointGrid at 0x1e68333a358>

In [68]:

```
sns.jointplot(x='TotalPayBenefits',y='OtherPay',data=sal,kind='hex',color='Red')
```

Out[68]:

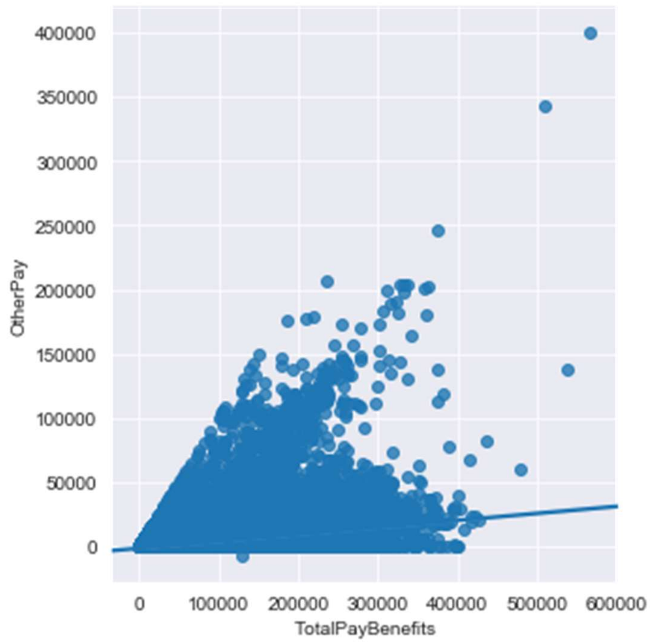
<seaborn.axisgrid.JointGrid at 0x1e6869224e0>



In [122]:

```
sns.lmplot(x='TotalPayBenefits',y='OtherPay',data=sal)
```

Out[122]: <seaborn.axisgrid.FacetGrid at 0x1e68bb8b668>

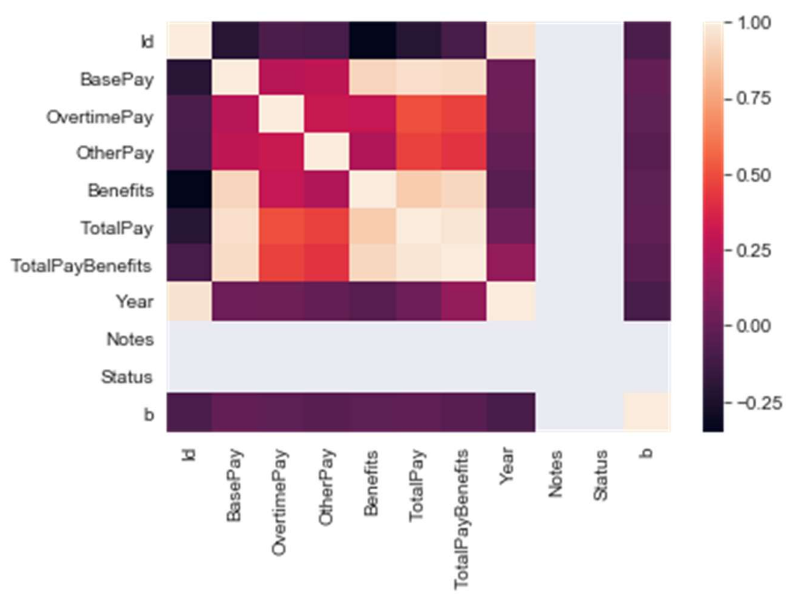


In [69]:

```
sns.heatmap(sal.corr())
```

Out[69]:

<matplotlib.axes._subplots.AxesSubplot at 0x1e686e2c438>

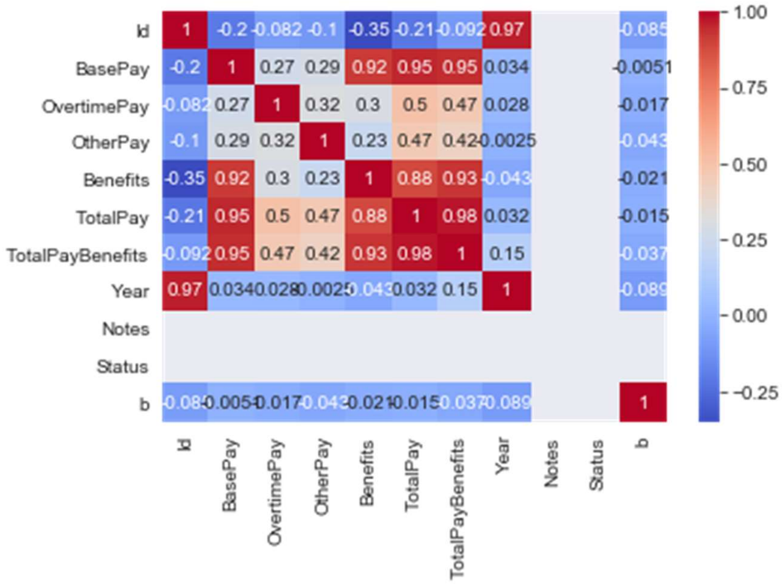


In [70]:

```
sns.heatmap(sal.corr(),cmap='coolwarm',annot=True)
```

Out[70]: <matplotlib.axes._subplots.AxesSubplot at

0x1e686ec0128>



In [71]:

```
USAhousing = pd.read_csv('USA_Housing.csv') #taking in a new dataframe because the furt
her operations cannot be carried out
                                                    #on previous dataframe due to its large si
ze
```

In [72]:

```
USAhousing.columns
```

Out[72]:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Roo
ms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Addres
s'],      dtype='object')
```

In [88]:

```
x = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
               'Avg. Area Number of Bedrooms', 'Area Population']]
y = USAhousing['Price']
```

In [89]:

```
from sklearn.model_selection import train_test_split
```

In [90]:

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

In [91]: from sklearn.linear_model import

```
```

LinearRegression In [92]:

```
lm=LinearRegression()
```

In [93]:

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

Out[93]:

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,

normalize=False) In [94]:

```
print(lm.intercept_)
```

-2640159.796851911 In

[95]:

```
cdf=pd.DataFrame(lm.coef_,X.columns,columns=['coeff'])
cdf
```

Out[95]:

	coeff
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

In [96]:

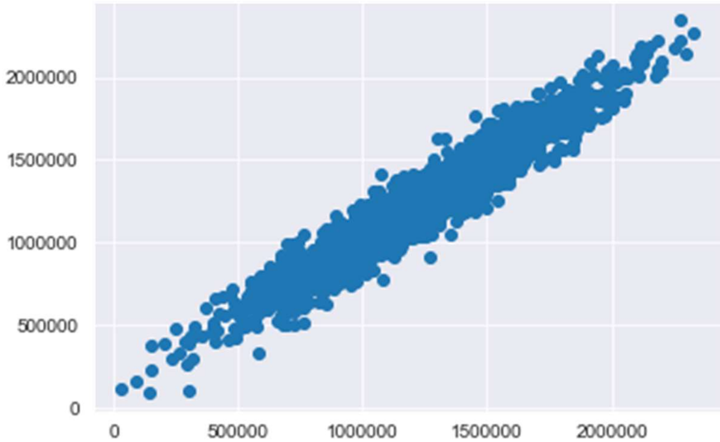
```
predictions=lm.predict(X_test)
```

In [97]:

plt.scatter(y_test,predictions)

Out[97]: <matplotlib.collections.PathCollection at

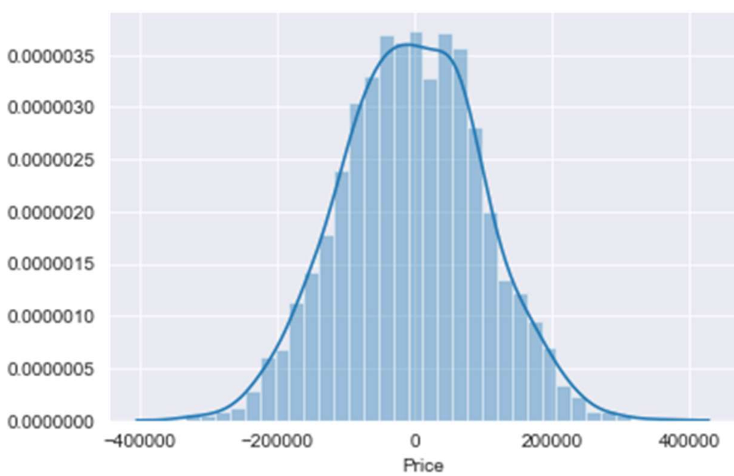
0x1e68ba02780>



In [98]:

```
sns.distplot((y_test-predictions))
```

Out[98]: <matplotlib.axes._subplots.AxesSubplot at 0x1e68ba31438>



In [99]:

```
from sklearn import metrics
```

In [100]:

```
a=metrics.mean_absolute_error(y_test,predictions)
b=metrics.mean_squared_error(y_test,predictions)
c=np.sqrt(metrics.mean_squared_error(y_test,predictions))
print('MAE:',a)
print('MSE:',b)
print('RMSE',c)
```

```
MAE: 82288.22251914957
MSE: 10460958907.209507
RMSE 102278.82922291156
```

In [109]:

```
ad_data=pd.read_csv('advertising.csv') #again taking new datframe for logistic regressi
ons because previous dataframe won't fit
```

In [110]:

```
ad_data.columns
```

Out[110]:

```
Index(['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Ad Topic Line', 'City', 'Male', 'Country',
       'Timestamp', 'Clicked on Ad'],
 dtype='object')
```

In [111]:

```
x=ad_data[['Daily Time Spent on Site','Age','Area Income','Daily Internet Usage','Male'
]]
y=ad_data['Clicked on Ad']
```

In [112]:

```
from sklearn.model_selection import train_test_split
```

In [113]:

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

In [114]:

```
from sklearn.linear_model import LogisticRegression
```

In [115]:

```
logmodel=LogisticRegression()
logmodel.fit(x_train,y_train)
```

```
C:\Users\ADHYAYAN SRIJAN\Anaconda3\lib\site-packages\sklearn\linear_model
\logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.  FutureWarning)
```

Out[115]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
```

```
         intercept_scaling=1, max_iter=100, multi_class='warn',
 n_jobs=None, penalty='l2', random_state=None, solver='warn',
 tol=0.0001, verbose=0, warm_start=False)
```

In [116]:

```
predictions=logmodel.predict(x_test)
```

In [117]:

```
from sklearn.metrics import confusion_matrix
```

In [118]:

```
print(confusion_matrix(y_test,predictions))
```

```
[[192  14]
 [ 24 170]] In
```

[119]:

```
from sklearn.metrics import classification_report
```

In [120]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support
0       0.89      0.93      0.91       206
1       0.92      0.88      0.90       194

   micro avg       0.91      0.91      0.91       400
macro avg       0.91      0.90      0.90       400 weighted
avg       0.91      0.91      0.90       400
 In
```

[124]:

```
seed=pd.read_csv('Seed_Data.csv')
```

In [125]:

```
seed.columns
```

Out[125]:

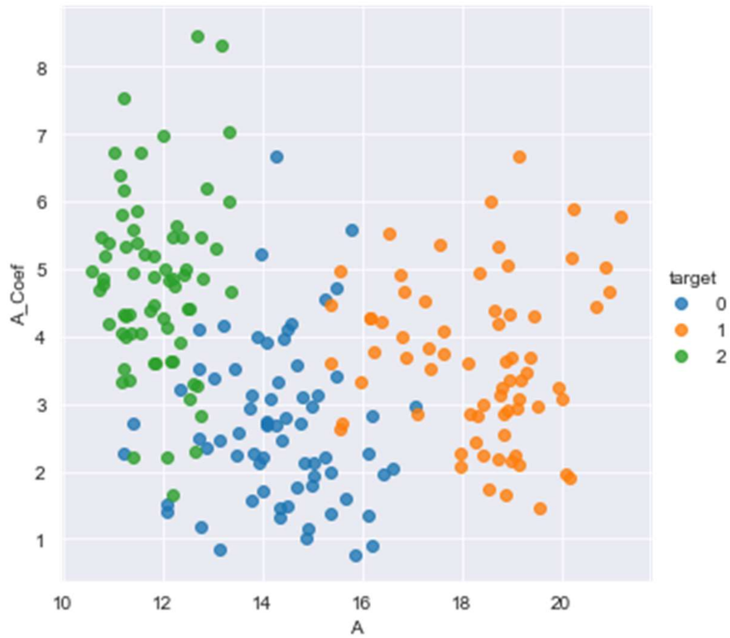
```
Index(['A', 'P', 'C', 'LK', 'WK', 'A_Coef', 'LKG', 'target'], dtype='objec
t')
```

In [126]:

```
sns.lmplot(x='A',y='A_Coef',data=a,hue='target',fit_reg=False)
```

Out[126]:

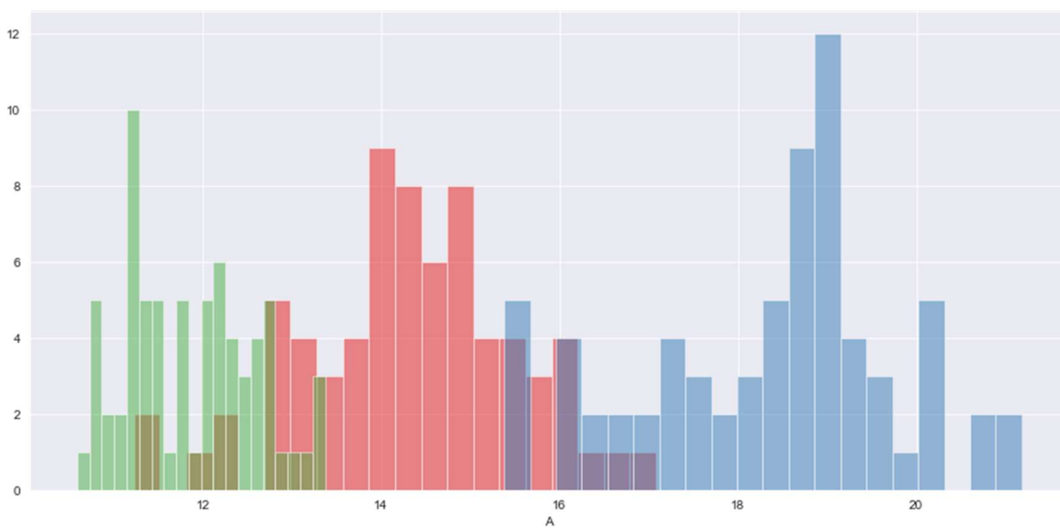
```
<seaborn.axisgrid.FacetGrid at 0x1e68be247f0>
```



In [127]:

```
g=sns.FacetGrid(a,hue='target',palette='Set1',size=6,aspect=2)
g=g.map(plt.hist,'A',bins=20,alpha=0.5)
```

```
C:\Users\ADHYAYAN SRIJAN\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2
30: UserWarning: The `size` paramter has been renamed to `height`; please
update your code.  warnings.warn(msg, UserWarning)
```



In [128]:

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

In [129]:

```
km=KMeans(n_clusters=3)
```

In [130]:

```
km.fit(a.drop('target',axis=1))
```

Out[130]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
random_state=None, tol=0.0001, verbose=0)
```

In [132]:

```
cen=km.cluster_centers_
cen
```

Out[132]:

```
array([[14.64847222, 14.46041667,  0.87916667,  5.56377778,  3.27790278,
2.64893333,  5.19231944],
       [18.72180328, 16.29737705,  0.88508689,  6.20893443,  3.72267213,
3.60359016,  6.06609836],
       [11.96441558, 13.27480519,  0.8522    ,  5.22928571,  2.87292208,
         4.75974026,  5.08851948]])
```

In [133]:

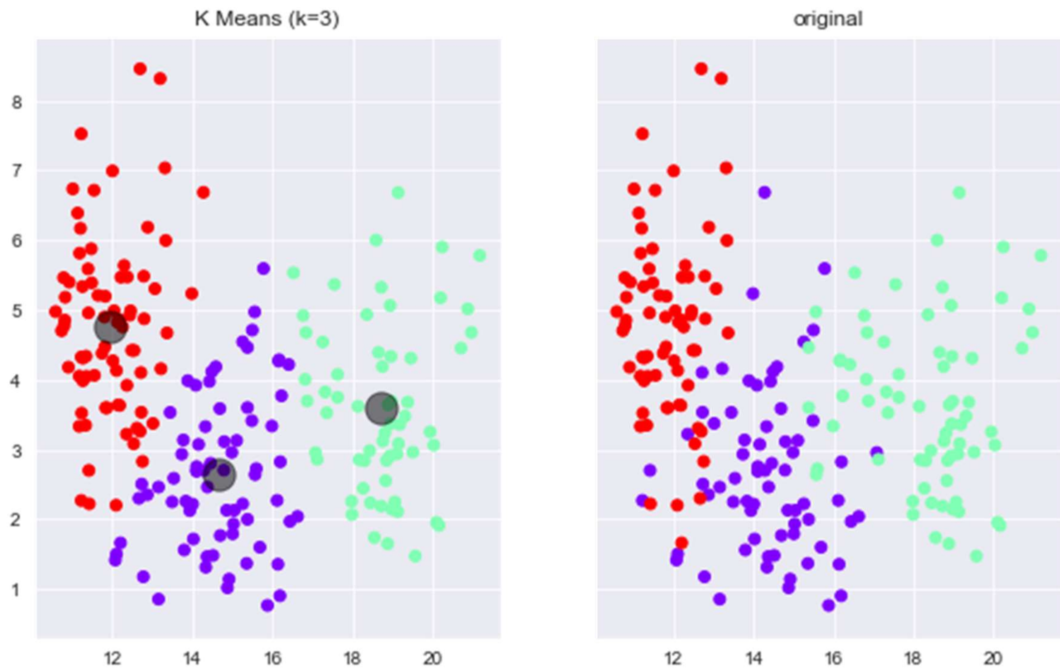
```
a['klabels']=km.labels_
a.head()
```

Out[133]:

	A	P	C	LK	WK	A_Coef	LKG	target	klabels
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	0	0
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	0	0
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	0	0
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	0	0
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	0	0

In [134]:

```
f,(ax1,ax2)=plt.subplots(nrows=1,ncols=2,sharey=True,figsize=(10,6))
ax1.set_title('K Means (k=3)')
ax1.scatter(x=a['A'],y=a['A_Coef'],c=a['klabels'],cmap='rainbow')
ax2.set_title('original')
ax2.scatter(x=a['A'],y=a['A_Coef'],c=a['target'],cmap='rainbow')
ax1.scatter(x=cen[:,0],y=cen[:,5],c='black',s=300,alpha=0.5);
```



In [135]:

```
sum_square={}
for k in range (1,10):
    kme=KMeans(n_clusters=k).fit(a)
    sum_square[k]=kme.inertia_
sum_square
```

Out[135]:

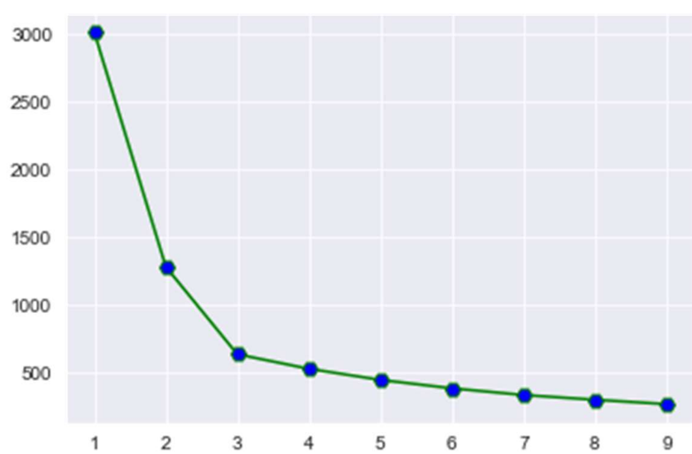
```
{1: 3008.7333625589044,
 2: 1280.2671443747977,
 3: 635.3722037214054,
 4: 528.5967154071016,
 5: 446.79384689994174,
6: 384.5568266088774,
 7: 335.50344030243826,
 8: 300.40119976428093,
 9: 268.3648997504013}
```

In [136]:

```
plt.plot(list(sum_square.keys()),list(sum_square.values()),ls='-',marker='H',color='g',
markersize=8,markerfacecolor='b')
```

Out[136]: [<matplotlib.lines.Line2D at

0x1e68b96c160>]



In [ ]:

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 complexity 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 will rise 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

✓ Coding : Self-performed