In [47]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
import statsmodels
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import statsmodels.formula.api as smf
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [48]:
```
# Importing the data:

expected_ctc= pd.read_csv('expected_ctc.csv')
expected_ctc.head(10)
```

Out[48]:

	IDX	Applicant_ID	Total_Experience	Total_Experience_in_field_applied	Department	
0	1	22753	0	0	NaN	
1	2	51087	23	14	HR	Cons
2	3	38413	21	12	Top Management	Cons
3	4	11501	15	8	Banking	Fin A
4	5	58941	10	5	Sales	P Ma
5	6	30564	16	3	Top Management	Area Ma
6	7	27267	1	1	Engineering	
7	8	36521	19	11	Others	A
8	9	11616	8	7	Analytics/BI	C
9	10	43886	15	15	Analytics/BI	

In [49]:
```
# Accomodating all columns on screen:

pd.options.display.max_columns = None
```

In [4]:
```
expected_ctc.head(10)
```

Out[4]:

	IDX	Applicant_ID	Total_Experience	Total_Experience_in_field_applied	Department	
0	1	22753	0	0	NaN	
1	2	51087	23	14	HR	Cons
2	3	38413	21	12	Top Management	Cons
3	4	11501	15	8	Banking	Fina Ar
4	5	58941	10	5	Sales	P Ma
5	6	30564	16	3	Top Management	Area Ma
6	7	27267	1	1	Engineering	
7	8	36521	19	11	Others	Ar
8	9	11616	8	7	Analytics/BI	C
9	10	43886	15	15	Analytics/BI	

In [5]:
```
### Detailed EDA has already been done in Project Notes-1, however,
```

In [5]:
```
# Checking the total entries:

expected_ctc.size
```

Out[5]: 725000

In [50]:
```
# Checking the data structure:

expected_ctc.shape
```

Out[50]: (25000, 29)

In [6]:
```
# Checking the data types:

expected_ctc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 29 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   IDX                               25000 non-null  int64  
 1   Applicant_ID                      25000 non-null  int64  
 2   Total_Experience                  25000 non-null  int64  
 3   Total_Experience_in_field_applied 25000 non-null  int64  
 4   Department                        22222 non-null  object 
 5   Role                              24037 non-null  object 
 6   Industry                          24092 non-null  object 
 7   Organization                      24092 non-null  object 
 8   Designation                       21871 non-null  object 
 9   Education                         25000 non-null  object 
 10  Graduation_Specialization         18820 non-null  object 
 11  University_Grad                   18820 non-null  object 
 12  Passing_Year_Of_Graduation        18820 non-null  float64
 13  PG_Specialization                 17308 non-null  object 
 14  University_PG                     17308 non-null  object 
 15  Passing_Year_Of_PG                17308 non-null  float64
 16  PHD_Specialization                13119 non-null  object 
 17  University_PHD                    13119 non-null  object 
 18  Passing_Year_Of_PHD               13119 non-null  float64
 19  Curent_Location                   25000 non-null  object 
 20  Preferred_location                25000 non-null  object 
 21  Current_CTC                       25000 non-null  int64  
 22  Inhand_Offer                      25000 non-null  object 
 23  Last_Appraisal_Rating             24092 non-null  object 
 24  No_Of_Companies_worked            25000 non-null  int64  
 25  Number_of_Publications            25000 non-null  int64  
 26  Certifications                    25000 non-null  int64  
 27  International_degree_any          25000 non-null  int64  
 28  Expected_CTC                      25000 non-null  int64  
dtypes: float64(3), int64(10), object(16)
memory usage: 5.5+ MB
```

In [51]:
```
#Dropping unnecessary columns that are unique identifiers and do no

expected_ctc=expected_ctc.drop(columns=["IDX","Applicant_ID"])
```

In [8]:
```
expected_ctc.head()
```

Out[8]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	O
0	0	0	NaN	NaN	NaN	
1	23	14	HR	Consultant	Analytics	
2	21	12	Top Management	Consultant	Training	
3	15	8	Banking	Financial Analyst	Aviation	
4	10	5	Sales	Project Manager	Insurance	

In [52]:
```
expected_ctc.shape
```

Out[52]: (25000, 27)

In [53]:
```
# Converting year_of_passing variables to categorical type:

expected_ctc['Passing_Year_Of_Graduation']=expected_ctc['Passing_Ye
expected_ctc['Passing_Year_Of_PG']=expected_ctc['Passing_Year_Of_PG
expected_ctc['Passing_Year_Of_PHD']=expected_ctc['Passing_Year_Of_PI
expected_ctc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 27 columns):
 #   Column                           Non-Null Count  Dtype 
---  ------                           --------------  ----- 
 0   Total_Experience                 25000 non-null  int64 
 1   Total_Experience_in_field_applied  25000 non-null  int64 
 2   Department                       22222 non-null  object
 3   Role                             24037 non-null  object
 4   Industry                         24092 non-null  object
 5   Organization                     24092 non-null  object
 6   Designation                      21871 non-null  object
 7   Education                        25000 non-null  object
 8   Graduation_Specialization        18820 non-null  object
 9   University_Grad                  18820 non-null  object
 10  Passing_Year_Of_Graduation       18820 non-null  object
 11  PG_Specialization                17308 non-null  object
 12  University_PG                    17308 non-null  object
 13  Passing_Year_Of_PG               17308 non-null  object
 14  PHD_Specialization               13119 non-null  object
 15  University_PHD                   13119 non-null  object
 16  Passing_Year_Of_PHD              13119 non-null  object
 17  Curent_Location                  25000 non-null  object
 18  Preferred_location               25000 non-null  object
 19  Current_CTC                      25000 non-null  int64 
 20  Inhand_Offer                     25000 non-null  object
 21  Last_Appraisal_Rating            24092 non-null  object
 22  No_Of_Companies_worked           25000 non-null  int64 
 23  Number_of_Publications           25000 non-null  int64 
 24  Certifications                   25000 non-null  int64 
 25  International_degree_any         25000 non-null  int64 
 26  Expected_CTC                     25000 non-null  int64 
dtypes: int64(8), object(19)
memory usage: 5.1+ MB
```

In [54]:
```
# Since 'Total_Experience_in_field_applied' showed presence of utli

def remove_outlier(col):
    sorted(col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    lower_range= Q1-(1.5 * IQR)
    upper_range= Q3+(1.5 * IQR)
    return lower_range, upper_range
```

In [55]:
```
lr,ur=remove_outlier(expected_ctc['Total_Experience_in_field_applie
expected_ctc['Total_Experience_in_field_applied']=np.where(expected_
expected_ctc['Total_Experience_in_field_applied']=np.where(expected_
```

In [56]:
```
## NaN value treatments:

# In columns - ""Industry","Last_Appraisal_Rating","Organization","

cat=["Industry","Last_Appraisal_Rating","Organization","Department"

for column in expected_ctc[cat]:
    expected_ctc[column]=expected_ctc[column].fillna("None")
```

In [57]:
```
## Similarly in columns - 'Graduation_Specialization','University_G

cat=['Graduation_Specialization','University_Grad','PG_Specializati

for column in expected_ctc[cat]:
    expected_ctc[column]=expected_ctc[column].fillna("Not_Applicabl
```

In [14]:
```
# Checking if NaN are replaced correctly:

expected_ctc.head(20)
```

Out[14]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry
0	0	0.0	None	None	None
1	23	14.0	HR	Consultant	Analytics
2	21	12.0	Top Management	Consultant	Training
3	15	8.0	Banking	Financial Analyst	Aviation
4	10	5.0	Sales	Project Manager	Insurance
5	16	3.0	Top Management	Area Sales Manager	Retail
6	1	1.0	Engineering	Team Lead	FMCG
7	19	11.0	Others	Analyst	Others
8	8	7.0	Analytics/BI	Others	Telecom
9	15	15.0	Analytics/BI	CEO	Telecom
10	13	10.0	Education	Business Analyst	Automobile
11	7	1.0	Marketing	Sales Manager	FMCG
12	10	10.0	Others	Bio statistician	Automobile
13	0	0.0	None	None	None
14	12	9.0	Banking	Bio statistician	Telecom
15	20	15.0	Healthcare	Analyst	IT
16	4	4.0	Analytics/BI	Scientist	Analytics
17	21	7.0	Healthcare	Research Scientist	BFSI
18	14	9.0	Sales	Business Analyst	Telecom
19	8	3.0	Engineering	Consultant	Telecom

In [18]:
```
### Converting discrete categorical to discrete numerical variables
```

In [58]:
```
# Converting 'Inhand_Offer' to boolean values:

expected_ctc['Inhand_Offer'].replace(['N','Y'],[0,1],inplace=True )
```

In [59]:
```
expected_ctc['Inhand_Offer'].unique()
```

Out[59]:
```
array([0, 1])
```

In [21]:
```
# There seem to be errors in the 'Education' column, since it is no
# Assuming that 'Education' has errors in it, we can create a new co
```

In [60]:
```
conditions=[
    (expected_ctc['University_PHD']=='Not_Applicable') & (expected_
    (expected_ctc['University_PHD']=='Not_Applicable') & (expected_
    (expected_ctc['University_PHD']=='Not_Applicable') & (expected_
    (expected_ctc['University_PHD']!='Not_Applicable') & (expected_
    ]

values=['Under_Grad','Graduate','Post_Grad','Doctorate']
```

In [61]:
```
expected_ctc['Edu_qualification']=np.select(conditions,values)
expected_ctc.head(25)
```

Out[61]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry
0	0	0.0	None	None	None
1	23	14.0	HR	Consultant	Analytics
2	21	12.0	Top Management	Consultant	Training
3	15	8.0	Banking	Financial Analyst	Aviation
4	10	5.0	Sales	Project Manager	Insurance
5	16	3.0	Top Management	Area Sales Manager	Retail
6	1	1.0	Engineering	Team Lead	FMCG
7	19	11.0	Others	Analyst	Others
8	8	7.0	Analytics/BI	Others	Telecom
9	15	15.0	Analytics/BI	CEO	Telecom
10	13	10.0	Education	Business Analyst	Automobile
11	7	1.0	Marketing	Sales Manager	FMCG

12	10	10.0	Others	Bio statistician	Automobile
13	0	0.0	None	None	None
14	12	9.0	Banking	Bio statistician	Telecom
15	20	15.0	Healthcare	Analyst	IT
16	4	4.0	Analytics/BI	Scientist	Analytics
17	21	7.0	Healthcare	Research Scientist	BFSI
18	14	9.0	Sales	Business Analyst	Telecom
19	8	3.0	Engineering	Consultant	Telecom
20	17	12.0	HR	Others	Training
21	7	6.0	Banking	Analyst	Training
22	22	6.0	Top Management	Consultant	BFSI
23	15	10.0	Sales	Head	Insurance
24	3	2.0	Banking	Associate	Aviation

In [62]:
```
# 'Education' column can now be dropped:

expected_ctc=expected_ctc.drop(columns=["Education"])
```

In [20]:
```
expected_ctc.head()
```

Out[20]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	O
0	0	0.0	None	None	None	
1	23	14.0	HR	Consultant	Analytics	
2	21	12.0	Top Management	Consultant	Training	
3	15	8.0	Banking	Financial Analyst	Aviation	
4	10	5.0	Sales	Project Manager	Insurance	

In [63]:
```
expected_ctc['Edu_qualification'].unique()
```

Out[63]:
```
array(['Graduate', 'Doctorate', 'Under_Grad', 'Post_Grad'], dtype=
object)
```

In [64]:
```
# Ordinal encoding for 'Edu_qualification', since EDA has shown a d

scores={"Under_Grad":0,"Graduate":1,"Post_Grad":2,"Doctorate":3}
expected_ctc['Edu_qualification']=expected_ctc['Edu_qualification']
```

In [65]:
```
expected_ctc['Edu_qualification'].unique()
```

Out[65]:
```
array([1, 3, 0, 2])
```

In [24]:
```
expected_ctc.head(10)
```

Out[24]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	O
0	0	0.0	None	None	None	
1	23	14.0	HR	Consultant	Analytics	
2	21	12.0	Top Management	Consultant	Training	
3	15	8.0	Banking	Financial Analyst	Aviation	
4	10	5.0	Sales	Project Manager	Insurance	
5	16	3.0	Top Management	Area Sales Manager	Retail	
6	1	1.0	Engineering	Team Lead	FMCG	
7	19	11.0	Others	Analyst	Others	
8	8	7.0	Analytics/BI	Others	Telecom	
9	15	15.0	Analytics/BI	CEO	Telecom	

In [25]:
```
expected_ctc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 27 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Total_Experience                  25000 non-null  int64  
 1   Total_Experience_in_field_applied 25000 non-null  float64
 2   Department                        25000 non-null  object 
 3   Role                              25000 non-null  object 
 4   Industry                          25000 non-null  object 
 5   Organization                      25000 non-null  object 
 6   Designation                       25000 non-null  object 
 7   Graduation_Specialization         25000 non-null  object 
 8   University_Grad                   25000 non-null  object 
 9   Passing_Year_Of_Graduation        18820 non-null  object 
 10  PG_Specialization                 25000 non-null  object 
 11  University_PG                     25000 non-null  object 
 12  Passing_Year_Of_PG                17308 non-null  object 
 13  PHD_Specialization                25000 non-null  object 
 14  University_PHD                    25000 non-null  object 
 15  Passing_Year_Of_PHD               13119 non-null  object 
 16  Curent_Location                   25000 non-null  object 
 17  Preferred_location                25000 non-null  object 
 18  Current_CTC                       25000 non-null  int64  
 19  Inhand_Offer                      25000 non-null  int64  
 20  Last_Appraisal_Rating             25000 non-null  object 
 21  No_Of_Companies_worked            25000 non-null  int64  
 22  Number_of_Publications            25000 non-null  int64  
 23  Certifications                    25000 non-null  int64  
 24  International_degree_any          25000 non-null  int64  
 25  Expected_CTC                      25000 non-null  int64  
 26  Edu_qualification                 25000 non-null  int64  
dtypes: float64(1), int64(9), object(17)
memory usage: 5.1+ MB
```

In [31]:
```
# For data preprocessing, dummy encoding of catrgorical variables w
```

In [66]:
```
## Changing city names to tiers to reduce dimensionality post encod

expected_ctc.replace(dict.fromkeys(['Bangalore','Chennai','Hyderaba
```

In [67]:
```
expected_ctc.replace(dict.fromkeys(['Mangalore','Jaipur','Bhubanesw
```

In [28]:
```
expected_ctc.head(10)
```

Out[28]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	O
0	0	0.0	None	None	None	
1	23	14.0	HR	Consultant	Analytics	
2	21	12.0	Top Management	Consultant	Training	
3	15	8.0	Banking	Financial Analyst	Aviation	
4	10	5.0	Sales	Project Manager	Insurance	
5	16	3.0	Top Management	Area Sales Manager	Retail	
6	1	1.0	Engineering	Team Lead	FMCG	
7	19	11.0	Others	Analyst	Others	
8	8	7.0	Analytics/BI	Others	Telecom	
9	15	15.0	Analytics/BI	CEO	Telecom	

In [68]:
```
# Checking for unique values:

columns=["Curent_Location","Preferred_location","University_PHD","U

for column in expected_ctc[columns]:
    print(column.upper(),': ',expected_ctc[column].nunique())
    print(expected_ctc[column].value_counts().sort_values(ascending
    print('\n')
```

```
CURENT_LOCATION :  2
Tier-2    16631
Tier-1     8369
Name: Curent_Location, dtype: int64


PREFERRED_LOCATION :  2
Tier-2    16745
Tier-1     8255
Name: Preferred_location, dtype: int64


UNIVERSITY_PHD :  3
Not_Applicable    11881
Tier-2             8946
Tier-1             4173
Name: University_PHD, dtype: int64


UNIVERSITY_PG :  3
Tier-2            11981
Not_Applicable     7692
Tier-1             5327
Name: University_PG, dtype: int64


UNIVERSITY_GRAD :  3
Tier-2            13020
Not_Applicable     6180
Tier-1             5800
Name: University_Grad, dtype: int64
```

In [30]:
```
# Classifying years of graduation, PG and PHD into intervals so as

columns=["Passing_Year_Of_Graduation","Passing_Year_Of_PG","Passing_

for column in expected_ctc[columns]:
    print(column.upper(),': ',expected_ctc[column].nunique())
    print(expected_ctc[column].value_counts())
    print('\n')
```

```
2013.0    435
1988.0    372
2014.0    328
1987.0    252
2019.0    239
2015.0    223
2018.0    218
2020.0    217
2016.0    210
2017.0    204
1986.0    146
Name: Passing_Year_Of_Graduation, dtype: int64


PASSING_YEAR_OF_PG :  36
2011.0    816
2013.0    815
2010.0    774
2012.0    766
2009.0    759
```

In [69]:
```
bins=[1980,1985,1990,1995,2000,2005,2010,2015,2020,2025]

expected_ctc['Year_Graduation_bin']=pd.cut(x=expected_ctc['Passing_
expected_ctc['Year_PG_bin']=pd.cut(x=expected_ctc['Passing_Year_Of_
expected_ctc['Year_PHD_bin']=pd.cut(x=expected_ctc['Passing_Year_Of_
```

In [32]:
```
expected_ctc.head()
```

Out[32]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	O
0	0	0.0	None	None	None	
1	23	14.0	HR	Consultant	Analytics	
2	21	12.0	Top Management	Consultant	Training	
3	15	8.0	Banking	Financial Analyst	Aviation	
4	10	5.0	Sales	Project Manager	Insurance	

In [70]:
```
check_nan_in_data=expected_ctc.isnull().sum()
print(check_nan_in_data)
```

```
Total_Experience                      0
Total_Experience_in_field_applied     0
Department                            0
Role                                  0
Industry                              0
Organization                          0
Designation                           0
Graduation_Specialization             0
University_Grad                       0
Passing_Year_Of_Graduation         6180
PG_Specialization                     0
University_PG                         0
Passing_Year_Of_PG                 7692
PHD_Specialization                    0
University_PHD                        0
Passing_Year_Of_PHD               11881
Curent_Location                       0
Preferred_location                    0
Current_CTC                           0
Inhand_Offer                          0
Last_Appraisal_Rating                 0
No_Of_Companies_worked                0
Number_of_Publications                0
Certifications                        0
International_degree_any              0
Expected_CTC                          0
Edu_qualification                     0
Year_Graduation_bin                6180
Year_PG_bin                        7692
Year_PHD_bin                      11881
dtype: int64
```

In [71]:
```
# Replacing NaN values with 'Not Applicable', since here imputation

cat=['Passing_Year_Of_Graduation','Passing_Year_Of_PG','Passing_Yea
for column in expected_ctc[cat]:
    expected_ctc[column]=expected_ctc[column].fillna("Not_Applicabl
```

In [35]:
```
expected_ctc.head()
```

Out[35]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	O
0	0	0.0	None	None	None	
1	23	14.0	HR	Consultant	Analytics	
2	21	12.0	Top Management	Consultant	Training	
3	15	8.0	Banking	Financial Analyst	Aviation	
4	10	5.0	Sales	Project Manager	Insurance	

In [72]:
```
# Converting year_bin variables to categorical type:

expected_ctc['Year_Graduation_bin']=expected_ctc['Year_Graduation_b
expected_ctc['Year_PG_bin']=expected_ctc['Year_PG_bin'].astype("obj
expected_ctc['Year_PHD_bin']=expected_ctc['Year_PHD_bin'].astype("o
expected_ctc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 30 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Total_Experience                  25000 non-null  int64
 1   Total_Experience_in_field_applied 25000 non-null  float64
 2   Department                        25000 non-null  object
 3   Role                              25000 non-null  object
 4   Industry                          25000 non-null  object
 5   Organization                      25000 non-null  object
 6   Designation                       25000 non-null  object
 7   Graduation_Specialization         25000 non-null  object
 8   University_Grad                   25000 non-null  object
 9   Passing_Year_Of_Graduation        25000 non-null  object
 10  PG_Specialization                 25000 non-null  object
 11  University_PG                     25000 non-null  object
 12  Passing_Year_Of_PG                25000 non-null  object
 13  PHD_Specialization                25000 non-null  object
 14  University_PHD                    25000 non-null  object
 15  Passing_Year_Of_PHD               25000 non-null  object
 16  Curent_Location                   25000 non-null  object
 17  Preferred_location                25000 non-null  object
 18  Current_CTC                       25000 non-null  int64
 19  Inhand_Offer                      25000 non-null  int64
 20  Last_Appraisal_Rating             25000 non-null  object
 21  No_Of_Companies_worked            25000 non-null  int64
 22  Number_of_Publications            25000 non-null  int64
 23  Certifications                    25000 non-null  int64
 24  International_degree_any          25000 non-null  int64
 25  Expected_CTC                      25000 non-null  int64
 26  Edu_qualification                 25000 non-null  int64
 27  Year_Graduation_bin               18820 non-null  object
 28  Year_PG_bin                       17308 non-null  object
 29  Year_PHD_bin                      13119 non-null  object
dtypes: float64(1), int64(9), object(20)
memory usage: 5.7+ MB
```

In [73]:
```
cat=['Year_Graduation_bin','Year_PG_bin','Year_PHD_bin']
for column in expected_ctc[cat]:
    expected_ctc[column]=expected_ctc[column].fillna("Not_Applicabl
```

In [44]:
```
expected_ctc.head(10)
```

Out[44]:

	Total_Experience	Total_Experience_in_field_applied	Department	Role	Industry	O
0	0	0.0	None	None	None	
1	23	14.0	HR	Consultant	Analytics	
2	21	12.0	Top Management	Consultant	Training	
3	15	8.0	Banking	Financial Analyst	Aviation	
4	10	5.0	Sales	Project Manager	Insurance	
5	16	3.0	Top Management	Area Sales Manager	Retail	
6	1	1.0	Engineering	Team Lead	FMCG	
7	19	11.0	Others	Analyst	Others	
8	8	7.0	Analytics/BI	Others	Telecom	
9	15	15.0	Analytics/BI	CEO	Telecom	

In [74]:
```
check_nan_in_data=expected_ctc.isnull().sum()
print(check_nan_in_data)
```

```
Total_Experience                    0
Total_Experience_in_field_applied    0
Department                          0
Role                                0
Industry                            0
Organization                        0
Designation                         0
Graduation_Specialization           0
University_Grad                     0
Passing_Year_Of_Graduation          0
PG_Specialization                   0
University_PG                       0
Passing_Year_Of_PG                  0
PHD_Specialization                  0
University_PHD                      0
Passing_Year_Of_PHD                 0
Curent_Location                     0
Preferred_location                  0
Current_CTC                         0
Inhand_Offer                        0
Last_Appraisal_Rating               0
No_Of_Companies_worked              0
Number_of_Publications              0
Certifications                      0
International_degree_any            0
Expected_CTC                        0
Edu_qualification                   0
Year_Graduation_bin                 0
Year_PG_bin                         0
Year_PHD_bin                        0
dtype: int64
```

In [75]:
```
## Changing education specialization names to subject categories in

expected_ctc.replace(dict.fromkeys(['Chemistry','Zoology','Botony']
expected_ctc.replace(dict.fromkeys(['Mathematics','Statistics'],'Ma
expected_ctc.replace(dict.fromkeys(['Arts','Psychology','Sociology'
```

In [76]:
```
columns=["Graduation_Specialization","PG_Specialization","PHD_Speci

for column in expected_ctc[columns]:
    print(column.upper(),': ',expected_ctc[column].nunique())
    print(expected_ctc[column].value_counts())
    print('\n')
```

```
GRADUATION_SPECIALIZATION :  7
Not_Applicable     6180
Pure_sciences      5189
Arts_Humanities    5123
Maths_Stats        3413
Economics          1774
Engineering        1661
Others             1660
Name: Graduation_Specialization, dtype: int64


PG_SPECIALIZATION :  7
Not_Applicable     7692
Pure_sciences      4591
Arts_Humanities    4220
Maths_Stats        3439
Economics          1755
Engineering        1674
Others             1629
Name: PG_Specialization, dtype: int64


PHD_SPECIALIZATION :  7
Not_Applicable     11881
Pure_sciences       3445
Arts_Humanities     2913
Maths_Stats         2614
Others              1545
Economics           1343
Engineering         1259
Name: PHD_Specialization, dtype: int64
```

In [31]:
```
# Finally checking for unique categorical values: (to check data hy
categorical=['Department','Role','Industry','Organization','Designa

for column in expected_ctc[categorical]:
    print(column.upper(),': ',expected_ctc[column].nunique())
    print(expected_ctc[column].value_counts().sort_values(ascending
    print('\n')
```

```
None              2778
Marketing         2379
Analytics/BI      2096
Healthcare        2062
Others            2041
Sales             1991
HR                1988
Banking           1952
Education         1948
Engineering       1937
Top Management    1632
Accounts          1118
IT-Software       1078
Name: Department, dtype: int64


ROLE :  25
Others                    2248
Bio statistician          1913
Analyst                   1892
```

In [ ]:

In [77]:
```
expected_ctc= pd.get_dummies(expected_ctc, columns=['Department','R
```

In [43]:
```
expected_ctc.head(10)
```

Out[43]:

	Total_Experience	Total_Experience_in_field_applied	Passing_Year_Of_Graduation	Passing_
0	0	0.0	2020.0	Nc
1	23	14.0	1988.0	
2	21	12.0	1990.0	
3	15	8.0	1997.0	
4	10	5.0	2004.0	
5	16	3.0	1998.0	
6	1	1.0	2011.0	
7	19	11.0	2001.0	Nc
8	8	7.0	2003.0	
9	15	15.0	1998.0	

In [78]:
```
# Columns 'Passing_Year_Of_Graduation','Passing_Year_Of_PG','Passin

expected_ctc=expected_ctc.drop(columns=['Passing_Year_Of_Graduation
```

In [79]:
```
expected_ctc.shape
```

Out[79]: (25000, 143)

In [80]:
```
expected_ctc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Columns: 143 entries, Total_Experience to Year_PHD_bin_Not_Applica
ble
dtypes: float64(1), int64(9), uint8(133)
memory usage: 5.1 MB
```

In [54]:
```
### Model building:
```

In [81]:
```
#Extracting the target column to create separate vectors for splitt

x = expected_ctc.drop("Expected_CTC", axis=1)
y = expected_ctc.pop("Expected_CTC")

x.head()
```

Out[81]:

	Total_Experience	Total_Experience_in_field_applied	Current_CTC	Inhand_Offer	No_Of_Co
0	0	0.0	0	0	
1	23	14.0	2702664	1	
2	21	12.0	2236661	1	
3	15	8.0	2100510	0	
4	10	5.0	1931644	0	

In [82]:
```
#Splitting the data into Training & Testing sets:

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size
```

In [83]:
```
#checking the dimensions of the two sets:

print('x_train',x_train.shape)
print('x_test',x_test.shape)
print('y_train',y_train.shape)
print('y_test',y_test.shape)
```

```
x_train (17500, 142)
x_test (7500, 142)
y_train (17500,)
y_test (7500,)
```

In [37]:
```
# Building a LinearRegression model and finding the bestfit model o

from sklearn.linear_model import LinearRegression

regression_model = LinearRegression()
regression_model.fit(x_train, y_train)
```

Out[37]:
```
LinearRegression()
```

In [59]:
```
#Checking the coefficients for each feature:

from sklearn import metrics

for idx, col_name in enumerate(x_train.columns):
    print("The coefficient for {} is {}".format(col_name, regressio
```

```
The coefficient for Role_Lab Executuve is 32775.6447403629
The coefficient for Role_None is 3753.64402971102
The coefficient for Role_Others is -56.60079629758957
The coefficient for Role_Principal Analyst is 6415.09437098778
The coefficient for Role_Professor is -909.9381943583301
The coefficient for Role_Project Manager is -2288.1258282706535
The coefficient for Role_Research Scientist is 85909.96521027279
The coefficient for Role_Researcher is -20012.551614403987
The coefficient for Role_Sales Execituve is 1307.9036347681822
The coefficient for Role_Sales Manager is -520.8415145142062
The coefficient for Role_Scientist is 5298.277142100883
The coefficient for Role_Senior Analyst is 13093.274674785851
The coefficient for Role_Senior Researcher is -1059.3391745529116
The coefficient for Role_Sr. Business Analyst is 9567.016430777692
The coefficient for Role_Team Lead is -460.2775162408675
The coefficient for Industry_Automobile is 3835.229465967636
The coefficient for Industry_Aviation is 2307.804833008578
The coefficient for Industry_BFSI is 5590.428636728054
The coefficient for Industry_FMCG is 1904.15105151379
The coefficient for Industry_IT is -40.028079033623044
```

In [51]:
```
#Finding the intercept value:

intercept = regression_model.intercept_

print("The intercept for our model is {}".format(intercept))
```

```
The intercept for our model is 83923.89081323612
```

In [52]:
```
# Calculating R2:

r2_train=regression_model.score(x_train, y_train)
print("R2 on training data is {}".format(r2_train))
#93% of variation in price is explained by predictors in training s

#Calculating R2 on test data:
r2_test=regression_model.score(x_test, y_test)
print("R2 on test data is {}".format(r2_test))
```

```
R2 on training data is 0.9928744656374558
R2 on test data is 0.9930025402502967
```

In [84]:
```
from sklearn import metrics
```

In [55]:
```
# Calculating RSME:

predicted_train=regression_model.fit(x_train, y_train).predict(x_tr
rsme_train=np.sqrt(metrics.mean_squared_error(y_train,predicted_tra
print("RSME on training data is {}".format(rsme_train))

predicted_test=regression_model.fit(x_train, y_train).predict(x_tes
rsme_test=np.sqrt(metrics.mean_squared_error(y_test,predicted_test)
print("RSME on test data is {}".format(rsme_test))
```

```
RSME on training data is 97582.27324718762
RSME on test data is 97934.3355324096
```

In [ ]:

In [85]:
```
## Using statsmodel for further clarity:
# Combining x and y into a single dataframe:

data_train = pd.concat([x_train, y_train], axis=1)
data_test=pd.concat([x_test,y_test],axis=1)
data_train.head()
```

Out[85]:

	Total_Experience	Total_Experience_in_field_applied	Current_CTC	Inhand_Offer	No_C
4289	16	6.0	2599539	0	
19621	12	11.0	1590046	1	
14965	25	13.0	3641226	0	
12321	14	0.0	1567804	0	
6269	20	17.0	3344366	0	

In [41]:
```
data_train.shape
```

Out[41]: (17500, 143)

In [65]:
```
data_train.columns
```

Out[65]:
```
Index(['Total_Experience', 'Total_Experience_in_field_applied', 'C
urrent_CTC',
       'Inhand_Offer', 'No_Of_Companies_worked', 'Number_of_Public
ations',
       'Certifications', 'International_degree_any', 'Edu_qualific
ation',
       'Department_Analytics/BI',
       ...
       'Year_PG_bin_(2015, 2020]', 'Year_PG_bin_(2020, 2025]',
       'Year_PG_bin_Not_Applicable', 'Year_PHD_bin_(1995, 2000]',
       'Year_PHD_bin_(2000, 2005]', 'Year_PHD_bin_(2005, 2010]',
       'Year_PHD_bin_(2010, 2015]', 'Year_PHD_bin_(2015, 2020]',
       'Year_PHD_bin_Not_Applicable', 'Expected_CTC'],
      dtype='object', length=143)
```

In [86]:
```
data_train.rename(columns = {'Department_Analytics/BI':'Department_
```

In [67]:
```
import statsmodels.formula.api as smf
lm1 = smf.ols(formula= 'Expected_CTC ~ Total_Experience + Total_Exp
print(lm1.summary())
```

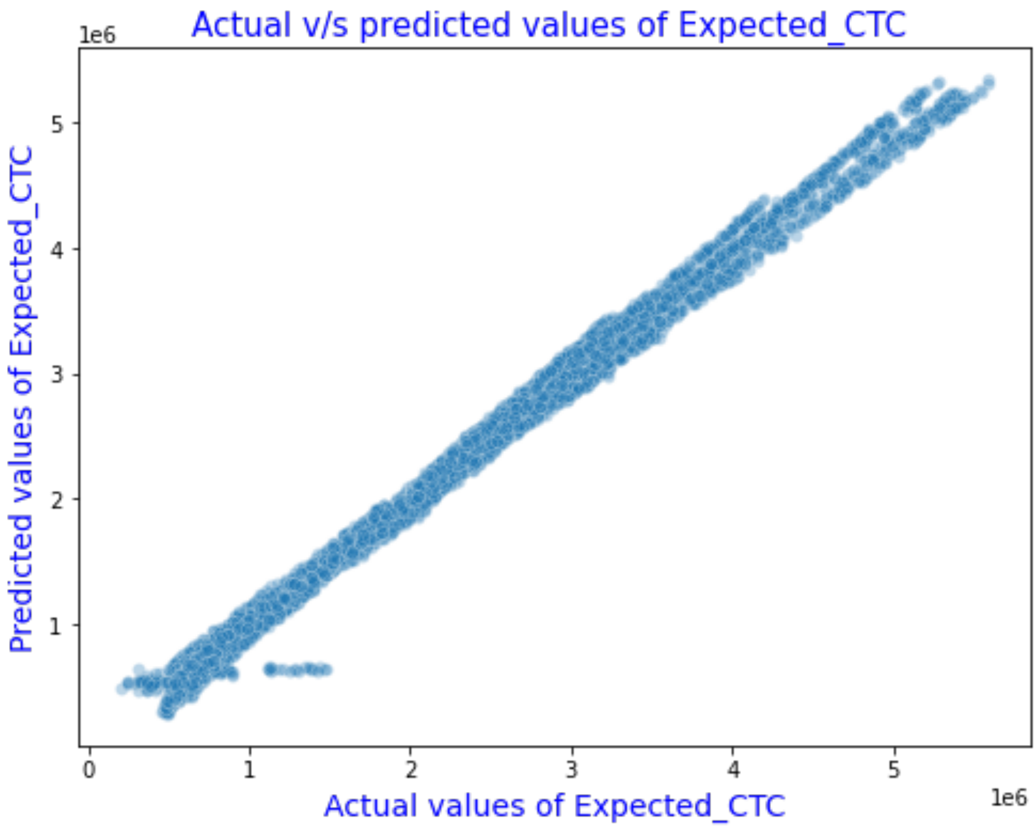
```
                            OLS Regression Results
==================================================================
============
Dep. Variable:           Expected_CTC   R-squared:
0.993
Model:                            OLS   Adj. R-squared:
0.993
Method:                 Least Squares   F-statistic:
1.819e+04
Date:                Sat, 14 May 2022   Prob (F-statistic):
0.00
Time:                        11:11:11   Log-Likelihood:
-2.2588e+05
No. Observations:               17500   AIC:
4.520e+05
Df Residuals:                   17366   BIC:
4.531e+05
Df Model:                         133
Covariance Type:            nonrobust
------------------------------------------------------------------
```

In [70]:
```
# Let us check the sum of squared errors by predicting value of y fo
# subtracting from the actual y for the test cases

mse = np.mean((regression_model.predict(x_test)-y_test)**2)
```

In [71]:
```
# underroot of mean_sq_error is standard deviation i.e. avg varianc

import math

math.sqrt(mse)

# so there is avg of 97934.335 (roundoff) mpg difference from real
```

Out[71]: 97934.3355324097

In [89]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(x=y_test, y=predicted_test, data=data_test, alpha=0
plt.title("Actual v/s predicted values of Expected_CTC ",color="blu
plt.xlabel("Actual values of Expected_CTC", color="blue",size=14)
plt.ylabel("Predicted values of Expected_CTC",color="blue",size=14)
plt.show()
```



In [ ]:
```
# Caculating VIF (Variable inflation Factor) to determine multicoll
```

In [68]:
```
def vif_cal(input_data):
    vars=input_data
    var_names=input_data.columns
    for i in range(0,var_names.shape[0]):
        y=vars[var_names[i]]
        x=vars[var_names.drop(var_names[i])]
        rsq=smf.ols(formula="y~x", data=vars).fit().rsquared
        vif=round(1/(1-rsq),2)
        print (var_names[i], " VIF = " , vif)
```

In [ ]:

In [69]:
```
vif_cal(input_data=data_train.drop('Expected_CTC',axis=1))
```

```
Total_Experience  VIF =  7.48
Total_Experience_in_field_applied  VIF =  1.74
Current_CTC  VIF =  5.42
Inhand_Offer  VIF =  1.62
No_Of_Companies_worked  VIF =  1.37
Number_of_Publications  VIF =  2.07
Certifications  VIF =  1.48
International_degree_any  VIF =  1.88

<ipython-input-68-006ae825bd87>:8: RuntimeWarning: divide by zero
encountered in double_scalars
  vif=round(1/(1-rsq),2)

Edu_qualification  VIF =  inf
Department_Analytics_BI  VIF =  3.03
Department_Banking  VIF =  2.68
Department_Education  VIF =  2.68
Department_Engineering  VIF =  2.62
Department_HR  VIF =  2.68
Department_Healthcare  VIF =  2.82
```

In [ ]:
```
### Iteration-2 --> dropping variable 'Total_Expeerience' which has
```

In [101]:
```
vif_cal(input_data=data_train.drop(['Expected_CTC','Total_Experienc
```

```
Total_Experience_in_field_applied  VIF =  1.57
Current_CTC  VIF =  3.57
Inhand_Offer  VIF =  1.62
No_Of_Companies_worked  VIF =  1.35
Number_of_Publications  VIF =  2.07
Certifications  VIF =  1.48
International_degree_any  VIF =  1.88

<ipython-input-68-006ae825bd87>:8: RuntimeWarning: divide by zero
encountered in double_scalars
  vif=round(1/(1-rsq),2)

Edu_qualification  VIF =  inf
Department_Analytics_BI  VIF =  3.02
Department_Banking  VIF =  2.67
Department_Education  VIF =  2.68
Department_Engineering  VIF =  2.61
Department_HR  VIF =  2.67
Department_Healthcare  VIF =  2.81
Department_IT_Software  VIF =  1.93
```

In [103]:
```
lm2 = smf.ols(formula= 'Expected_CTC ~ Total_Experience_in_field_ap
print(lm2.summary())
```

```
                            OLS Regression Results
==================================================================
============
Dep. Variable:            Expected_CTC   R-squared:
0.992
Model:                             OLS   Adj. R-squared:
0.992
Method:                  Least Squares   F-statistic:
1.726e+04
Date:                 Sat, 14 May 2022   Prob (F-statistic):
0.00
Time:                         13:45:08   Log-Likelihood:
-2.2640e+05
No. Observations:                17500   AIC:
4.531e+05
Df Residuals:                    17367   BIC:
4.541e+05
Df Model:                          132
Covariance Type:             nonrobust
```

In [ ]:
```
# In the 2nd model also, r^2 and adjussred R^2 is showing very mini
```

In [87]:
```
### Building other models (ANN, Decision tree and Random Forest) al

#As a prerequisite for ANN model, data will be scaled before model 

from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x_train_scaled=ss.fit_transform(x_train)
x_test_scaled=ss.transform(x_test)
```

In [78]:
```
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

ann = MLPRegressor(hidden_layer_sizes=(300),random_state=123, max_i
rf = RandomForestRegressor(random_state=123)
dt = tree.DecisionTreeRegressor(criterion='gini',random_state=123)
lr = LinearRegression()

models=[lr,dt,rf,ann]

rmse_train=[]
rmse_test=[]
scores_train=[]
scores_test=[]

for i in models:
    if (i != ann) :
        i.fit(x_train,y_train)
        scores_train.append(i.score(x_train, y_train))
        scores_test.append(i.score(x_test, y_test))
        rmse_train.append(np.sqrt(mean_squared_error(y_train,i.pred
        rmse_test.append(np.sqrt(mean_squared_error(y_test,i.predic

    else :
        i.fit(x_train_scaled,y_train)
        scores_train.append(i.score(x_train_scaled, y_train))
        scores_test.append(i.score(x_test_scaled, y_test))
        rmse_train.append(np.sqrt(mean_squared_error(y_train,i.pred
        rmse_test.append(np.sqrt(mean_squared_error(y_test,i.predic

print(pd.DataFrame({'Train RMSE': rmse_train,'Test RMSE': rmse_test
            index=['Linear Regression','Decision Tree Regressor','R
```

```
/Users/sabita/opt/anaconda3/lib/python3.8/site-packages/sklearn/ne
ural_network/_multilayer_perceptron.py:582: ConvergenceWarning: St
ochastic Optimizer: Maximum iterations (5000) reached and the opti
mization hasn't converged yet.
  warnings.warn(

                           Train RMSE     Test RMSE  Training Scor
e  \
```

```
Linear Regression        97582.273247  97934.335532        0.99287
4
Decision Tree Regressor  10608.936386  93445.970082        0.99991
6
Random Forest Regressor  28114.073619  69852.484628        0.99940
9
ANN Regressor            68880.980429  82933.139529        0.99645
0

                          Test Score
Linear Regression           0.993003
Decision Tree Regressor     0.993629
Random Forest Regressor     0.996440
ANN Regressor               0.994982
```

In [79]:
```
### Tuning the models with grid search to find the best parameters
```

In [42]:
```
from sklearn.model_selection import GridSearchCV
```

In [59]:
```
# Grid search CV for Decision Tree regressor:

param_grid = {
    'max_depth': [10,15,20,25,30],
    'min_samples_leaf': [3, 10,20],
    'min_samples_split': [10,25,30,40,50],
}

dtr=tree.DecisionTreeRegressor(random_state=123)

grid_search = GridSearchCV(estimator = dtr, param_grid = param_grid


grid_search.fit(x_train,y_train)

print(grid_search.best_params_)
```

```
{'max_depth': 20, 'min_samples_leaf': 10, 'min_samples_split': 10}
```

In [60]:
```
best_grid = grid_search.best_estimator_
best_grid
```

Out[60]:
```
DecisionTreeRegressor(max_depth=20, min_samples_leaf=10, min_sampl
es_split=10,
                      random_state=123)
```

In [70]:
```
# Generating the decision tree:

train_char_label = ['no', 'yes']
tree_regularized = open('tree_regularized.doc','w')
doc_data = tree.export_graphviz(best_grid, out_file= tree_regulariz
tree_regularized.close()
doc_data
```

In [62]:
```
# Genrating the decision tree on the site http://webgraphviz.com/
```

In [65]:
```
# Understanding feature importance:

pd.options.display.max_rows = None
print (pd.DataFrame(best_grid.feature_importances_, columns = ["Imp"
```

```
                                                 Imp
Current_CTC                             9.878782e-01
Last_Appraisal_Rating_D                 4.387552e-03
Last_Appraisal_Rating_C                 4.159063e-03
Total_Experience                        8.723766e-04
Edu_qualification                       6.599851e-04
Inhand_Offer                            6.560692e-04
Certifications                          3.051328e-04
Year_Graduation_bin_Not_Applicable      1.897096e-04
Graduation_Specialization_Not_Applicable 1.595508e-04
Number_of_Publications                  8.353873e-05
Last_Appraisal_Rating_B                 7.445832e-05
PHD_Specialization_Engineering          6.221591e-05
Total_Experience_in_field_applied       5.553028e-05
No_Of_Companies_worked                  4.361974e-05
PHD_Specialization_Maths_Stats          4.161556e-05
PG_Specialization_Pure_sciences         3.716847e-05
PG_Specialization_Maths_Stats           3.603037e-05
Last_Appraisal_Rating_Key_Performer     3.018223e-05
PG_Specialization_Others                2.550973e-05
```

In [66]:

In [83]:
```
# Grid search CV for Random Forest regressor:

param_grid = {
    'max_depth': [10,15],
    'max_features': [15, 20],
    'min_samples_leaf': [5, 15,30],
    'min_samples_split': [20,40,60],
    'n_estimators': [200, 300]
}

rfr = RandomForestRegressor(random_state=123)

grid_search = GridSearchCV(estimator = rfr, param_grid = param_grid
grid_search.fit(x_train,y_train)
```

Out[83]:
```
GridSearchCV(cv=3, estimator=RandomForestRegressor(random_state=12
3),
             param_grid={'max_depth': [10, 15], 'max_features': [1
5, 20],
                         'min_samples_leaf': [5, 15, 30],
                         'min_samples_split': [20, 40, 60],
                         'n_estimators': [200, 300]})
```

In [84]:
```
print(grid_search.best_params_)
```

```
{'max_depth': 15, 'max_features': 20, 'min_samples_leaf': 5, 'min_
samples_split': 20, 'n_estimators': 300}
```

In [ ]:

In [68]:
```
# Grid search CV for Artificial Neural Network regressor:

#from sklearn.neural_network import MLPRegressor
#param_grid = {
    'hidden_layer_sizes': [50,100],
    'max_iter': [5500,7000],
    'solver': ['adam','sgd'],
    'tol': [0.01],
}

#nnr = MLPRegressor(random_state=1)

#grid_search = GridSearchCV(estimator = nnr, param_grid = param_gri
```

In [43]:
```
#grid_search.fit(x_train_scaled, y_train)
#grid_search.best_params_
```

In [ ]:
```
#best_grid = grid_search.best_estimator_
#best_grid
```

In [38]:
```
### Creating regularised models using Lasso & Ridge:
```

In [45]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
```

In [58]:
```
ridge = Ridge(alpha=50)
ridge.fit(x_train,y_train)
print ("Ridge model:", (ridge.coef_))
```

```
Ridge model: [-8.48413668e+03  1.18324315e+02  1.34308566e+00  5.7
0663881e+04
 -6.20445524e+02  4.86656852e+02 -1.47168253e+04  8.17013395e+02
  9.82274079e+03 -1.36352392e+04 -1.55319171e+04  6.94969917e+03
 -1.41885568e+04 -1.44267457e+04 -1.18579162e+04 -1.02480824e+04
 -1.44679356e+04 -1.02068394e+04 -1.21891835e+04 -1.15131721e+04
 -2.04921908e+04  4.36571160e+03 -4.94057049e+03 -2.75344995e+03
  4.66309185e+02 -1.75912174e+03  1.52884180e+03  1.42080801e+03
  2.28774043e+03 -7.74967612e+03  7.83965976e+03  7.87113278e+04
 -3.15447730e+03  3.17903775e+03 -1.36268762e+03 -5.38419583e+03
  2.57190989e+04 -1.48431265e+04 -1.88324706e+03 -3.19135348e+03
  2.02501915e+03  5.30110379e+03 -4.78787825e+03  2.17277089e+03
 -3.66228802e+03 -1.40030036e+02 -1.99242847e+03  1.62049931e+03
 -2.02642127e+03 -3.75139161e+03 -2.05898446e+02  1.39019696e+05
  1.97135359e+03  1.75590808e+02 -2.51455781e+03 -1.75455199e+03
 -9.61802025e+03 -6.68355677e+03 -1.33553264e+03 -8.55296351e+03
 -1.36646873e+03 -8.77890010e+03 -8.85088236e+03 -2.87653623e+03
 -6.03578600e+03 -3.70486502e+03 -5.80906521e+03 -2.13598257e+03
 -4.08469731e+03  1.39019696e+05 -7.75075952e+03 -5.71812197e+03
 -9.24717647e+03 -3.14967224e+03  2.02324184e+03 -8.10970177e+03
 -9.05696113e+03 -1.92864467e+02 -6.08015705e+03 -1.33394790e+03
 -4.68991147e+03  2.65067562e+03 -8.08967473e+03 -4.46415338e+03
  5.00424703e+03 -1.29942936e+04  2.76321551e+01 -1.38090396e+03
 -2.71206812e+03 -3.00507037e+03 -2.56735707e+04 -1.98642221e+05
 -2.02526376e+05  1.69275460e+04  1.39019696e+05  5.50676700e+03
  1.17560046e+04 -7.81363170e+03 -5.87059083e+03  2.83805714e+03
  3.64364099e+03  8.54772689e+02  5.01581815e+03 -6.10039414e+02
 -3.31511917e+03 -1.27388522e+03 -3.97741869e+02 -1.57484449e+03
  2.28790666e+03  2.02983396e+03 -1.63209209e+03  4.03268256e+03
  7.78745260e+03  2.50125802e+04 -3.55440809e+03 -1.46912168e+03
 -8.40883732e+03  1.08594024e+03  2.46846785e+03 -4.30627471e+03
  2.43380266e+02 -3.50437147e+03 -4.71828944e+03 -3.00396208e+04
 -3.17941331e+04  1.73528126e+04 -3.27481548e+04 -5.87059083e+03
 -6.41463863e+03 -1.56960068e+04 -8.67543964e+03 -4.38040784e+03
 -1.02788521e+04  8.55604483e+03 -2.74297101e+03 -3.97741869e+02
 -6.60918586e+03 -3.25725688e+04 -2.79599353e+04  1.11765836e+04
  1.20119366e+04 -3.55440809e+03]
```

In [59]:
```
print(ridge.score(x_train, y_train))
print(ridge.score(x_test, y_test))
```

```
0.9928388928789039
0.9929874356258246
```

In [62]:
```
lasso = Lasso(alpha=50)
lasso.fit(x_train,y_train)
print ("Lasso model:", (lasso.coef_))
```

```
Lasso model: [-8.63276030e+03  1.14033545e+02  1.34377050e+00  5.8
3630625e+04
 -2.16567618e+02  4.40031297e+02 -1.45930789e+04  5.72122516e+02
  1.05844671e+04 -1.07437085e+04 -1.33487190e+04  8.64634187e+03
 -1.20660228e+04 -1.22718770e+04 -9.70871011e+03 -7.98585900e+03
 -1.01044662e+04 -1.25643441e+04 -9.91593140e+03 -9.28177588e+03
 -1.75540171e+04  0.00000000e+00 -8.54022699e+02 -3.40032526e+02
  1.65058236e+03 -0.00000000e+00  2.86829378e+03  0.00000000e+00
  3.30834079e+03 -4.96638459e+03  0.00000000e+00  1.62188766e+03
 -7.23085316e+02  9.23458697e+02 -0.00000000e+00 -2.81086299e+03
  4.63864094e+04 -1.08508406e+04  0.00000000e+00 -5.63465059e+02
  2.93326274e+03  0.00000000e+00 -1.69495933e+03  0.00000000e+00
 -1.03966896e+03  1.17156847e+03  0.00000000e+00  2.78275308e+03
 -0.00000000e+00 -1.56852947e+03  1.18637133e+03  5.17320602e+05
  3.28798525e+03  1.68197412e+03 -5.37698968e+01  0.00000000e+00
 -4.59018182e+03 -1.17892134e+03  2.37076936e+03 -3.30557545e+03
  2.52737926e+03 -3.43107261e+03 -3.68325573e+03  1.03744814e+03
 -6.23029363e+02  1.84430765e+02 -1.06744196e+02  1.91889329e+03
  0.00000000e+00  2.42606339e+03 -2.37828675e+03 -1.95036731e+02
 -5.94302994e+03 -3.67179255e+02  3.11403344e+03 -4.96436996e+03
 -6.80689771e+03  1.12724660e+03 -3.65604135e+03  0.00000000e+00
 -1.32757800e+03  0.00000000e+00 -5.42786655e+03 -2.18561448e+03
  6.25017124e+03 -1.12818294e+04  0.00000000e+00  0.00000000e+00
 -3.12373026e+02 -0.00000000e+00 -2.92649999e+04 -2.05061747e+05
 -2.09059182e+05  1.20626134e+04  1.03025072e+01  2.58795306e+03
  6.69946841e+03 -1.06388973e+04 -0.00000000e+00  0.00000000e+00
  2.73588597e+03 -0.00000000e+00  1.96865541e+03  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00 -4.68112348e+02
  1.40213768e+03  0.00000000e+00 -1.48708134e+03  4.01166144e+03
  7.89282724e+03  2.52382443e+04 -0.00000000e+00 -0.00000000e+00
 -6.88010324e+03 -0.00000000e+00  1.35723211e+03 -4.15074925e+03
  7.52038017e+01 -3.26271396e+03 -5.11703465e+03 -3.19194221e+04
 -3.31539324e+04  1.41612351e+04 -3.74293219e+04 -1.51753341e+04
 -3.21113356e+03 -9.90070514e+03 -1.58403485e+03 -0.00000000e+00
 -6.75833911e+03  9.52560118e+03 -0.00000000e+00  0.00000000e+00
 -1.08014766e+04 -4.07542547e+04 -3.62198104e+04  5.59384698e+03
  6.21328514e+03 -8.92182513e+03]
```

In [63]:
```
print(lasso.score(x_train, y_train))
print(lasso.score(x_test, y_test))
```

```
0.992858570356946
0.9930037340269746
```

In [ ]:
```
### Using polynomial feature expansion for further checking:
```

In [46]:
```
from sklearn import preprocessing
from sklearn.preprocessing import PolynomialFeatures
```

In [47]:
```
# Scaling all the columns before polynomial feature processing:

x_scaled = preprocessing.scale(x)
y_scaled = preprocessing.scale(y)
x_train_scaled, x_test_scaled, y_train_scaled, y_test_scaled = trai
```

In [48]:
```
poly = PolynomialFeatures(degree = 2, interaction_only=True)
```

In [49]:
```
x_poly = poly.fit_transform(x_scaled)
x_poly_train, x_poly_test, y_poly_train, y_poly_test = train_test_s
x_poly_train.shape
```

Out[49]:
```
(17500, 10154)
```

In [50]:
```
# Fitting a simple non-regularized linear model on poly features:

regression_model = LinearRegression()
regression_model.fit(x_poly_train, y_poly_train)
```

Out[50]:
```
LinearRegression()
```

In [51]:
```
ridge = Ridge(alpha=50)
ridge.fit(x_poly_train,y_poly_train)
```

Out[51]:
```
Ridge(alpha=50)
```

In [52]:
```
print(ridge.score(x_poly_train, y_poly_train))
print(ridge.score(x_poly_test, y_poly_test))
```

```
0.997468003090859
0.9904693141999761
```

In [53]:
```
lasso = Lasso(alpha=50)
lasso.fit(x_poly_train,y_poly_train)
```

```
/Users/sabita/opt/anaconda3/lib/python3.8/site-packages/sklearn/li
near_model/_coordinate_descent.py:529: ConvergenceWarning: Objecti
ve did not converge. You might want to increase the number of iter
ations. Duality gap: 9789786430576.95, tolerance: 2338635145561.99
1
  model = cd_fast.enet_coordinate_descent(
```

Out[53]:
```
Lasso(alpha=50)
```

In [54]:
```
print(lasso.score(x_poly_train, y_poly_train))
print(lasso.score(x_poly_test, y_poly_test))
```

```
0.9973917653305221
0.9944631786604274
```

In [57]:
```
lasso = Lasso(alpha=200)
lasso.fit(x_poly_train,y_poly_train)
```

```
/Users/sabita/opt/anaconda3/lib/python3.8/site-packages/sklearn/li
near_model/_coordinate_descent.py:529: ConvergenceWarning: Objecti
ve did not converge. You might want to increase the number of iter
ations. Duality gap: 3443908380766.547, tolerance: 2338635145561.9
91
  model = cd_fast.enet_coordinate_descent(
```

Out[57]:
```
Lasso(alpha=200)
```

In [58]:
```
print(lasso.score(x_poly_train, y_poly_train))
print(lasso.score(x_poly_test, y_poly_test))
```

```
0.9969722884295543
0.9953317792133534
```

In [ ]:
```
### Generating additional ensemble models:
```

In [72]:
```
# Creating model using Gradient Boosting (GB):

from sklearn.ensemble import GradientBoostingRegressor

gb = GradientBoostingRegressor(n_estimators = 50,random_state=1)
gb = gb.fit(x_train, y_train)
```

In [74]:
```
y_predict = gb.predict(x_test)
print(gb.score(x_train, y_train))
print(gb.score(x_test, y_test))
```

```
0.9942524461191704
0.9942424756733194
```

In [44]:
```
pip install -U xgboost --upgrade
```

```
Collecting xgboost
  Downloading xgboost-1.6.1-py3-none-macosx_10_15_x86_64.macosx_11
_0_x86_64.macosx_12_0_x86_64.whl (1.7 MB)
     |████████████████████████████████| 1.7 MB 2.6 MB/s eta 0:00:0
1
Requirement already satisfied, skipping upgrade: numpy in /Users/s
abita/opt/anaconda3/lib/python3.8/site-packages (from xgboost) (1.
22.3)
Requirement already satisfied, skipping upgrade: scipy in /Users/s
abita/opt/anaconda3/lib/python3.8/site-packages (from xgboost) (1.
5.0)
Installing collected packages: xgboost
Successfully installed xgboost-1.6.1
Note: you may need to restart the kernel to use updated packages.
```

In [90]:
```
# Renaming the variables again to enable XGBoosting:

x_train.rename(columns = {'Department_Analytics/BI':'Department_Ana
```

```
<ipython-input-90-8bcfd763a787>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy)
  x_train.rename(columns = {'Department_Analytics/BI':'Department_
Analytics_BI','Department_IT-Software':'Department_IT_Software','D
epartment_Top Management':'Department_Top_Management','Role_Area S
ales Manager':'Role_Area_Sales_Manager','Role_Bio statistician':'R
ole_Bio_statistician','Role_Business Analyst':'Role_Business_Analy
st','Role_Data scientist':'Role_Data_scientist','Role_Financial An
alyst':'Role_Financial_Analyst','Role_Lab Executuve':'Role_Lab_Exe
cutuve','Role_Principal Analyst':'Role_Principal_Analyst','Role_Pr
oject Manager':'Role_Project_Manager','Role_Research Scientist':'R
ole_Research_Scientist','Role_Sales Execituve':'Role_Sales_Executu
ve','Role_Sales Manager':'Role_Sales_Manager','Role_Senior Analyst
':'Role_Senior_Analyst','Role_Senior Researcher':'Role_Senior_Rese
archer','Role_Sr. Business Analyst':'Role_Sr_Business_Analyst','Ro
le_Team Lead':'Role_Team_Lead','Designation_Data Analyst':'Designa
tion_Data_Analyst','Designation_Marketing Manager':'Designation_Ma
```

```
rketing_Manager','Designation_Medical Officer':'Designation_Medica
l_Officer','Designation_Network Engineer':'Designation_Network_Eng
ineer','Designation_Product Manager':'Designation_Product_Manager'
,'Designation_Research Analyst':'Designation_Research_Analyst','De
signation_Research Scientist':'Designation_Research_Scientist','De
signation_Software Developer':'Designation_Software_Developer','De
signation_Sr.Manager':'Designation_Sr_Manager','Designation_Web De
signer':'Designation_Web_Designer','University_Grad_Tier-1':'Unive
rsity_Grad_Tier_1','University_Grad_Tier-2':'University_Grad_Tier_
2','University_PG_Tier-1':'University_PG_Tier_1','University_PG_Ti
er-2':'University_PG_Tier_2','University_PHD_Tier-1':'University_P
HD_Tier_1','University_PHD_Tier-2':'University_PHD_Tier_2','Curent
_Location_Tier-2':'Curent_Location_Tier_2','Preferred_location_Tie
r-2':'Preferred_location_Tier_2','Year_Graduation_bin_(1990, 1995]
':'Year_Graduation_bin_1990_1995','Year_Graduation_bin_(1995, 2000
]':'Year_Graduation_bin_1995_2000','Year_Graduation_bin_(2000, 200
5]':'Year_Graduation_bin_2000_2005','Year_Graduation_bin_(2005, 20
10]':'Year_Graduation_bin_2005_2010','Year_Graduation_bin_(2010, 2
015]':'Year_Graduation_bin_2010_2015','Year_Graduation_bin_(2015,
2020]':'Year_Graduation_bin_2015_2020','Year_PG_bin_(1990, 1995]':
'Year_PG_bin_1990_1995','Year_PG_bin_(1995, 2000]':'Year_PG_bin_19
95_2000','Year_PG_bin_(2000, 2005]':'Year_PG_bin_2000_2005','Year_
PG_bin_(2005, 2010]':'Year_PG_bin_2005_2010','Year_PG_bin_(2010, 2
015]':'Year_PG_bin_2010_2015','Year_PG_bin_(2015, 2020]':'Year_PG_
bin_2015_2020','Year_PG_bin_(2020, 2025]':'Year_PG_bin_2020_2025',
'Year_PHD_bin_(1995, 2000]':'Year_PHD_bin_1995_2000','Year_PHD_bin
_(2000, 2005]':'Year_PHD_bin_2000_2005','Year_PHD_bin_(2005, 2010]
':'Year_PHD_bin_2005_2010','Year_PHD_bin_(2010, 2015]':'Year_PHD_b
in_2010_2015','Year_PHD_bin_(2015, 2020]':'Year_PHD_bin_2015_2020'
},inplace=True)
```

In [98]:
```
# Creating model using Xtreme Gradient Boosting (XGB):

import xgboost as xgb

model=xgb.XGBRegressor(n_estimators=500, max_depth=7, eta=0.1, subsa
model.fit(x_train, y_train)
```

Out[98]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_by
tree=0.8,
             early_stopping_rounds=None, enable_categorical=False,
eta=0.1,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='de
pthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.100000001, max_bin=256, max_cat_to_on
ehot=4,
             max_delta_step=0, max_depth=7, max_leaves=0, min_chil
d_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=
500, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0
, reg_alpha=0, ...)
```

In [99]:
```
y_pred=model.predict(x_test)
model_score_train=model.score(x_train,y_train)
model_score_test=model.score(x_test,y_test)
print(model_score_train)
print(model_score_test)
```

```
0.999474549736053
0.99635932877079
```

In [101]:
```
### The End
```