**Department of Computer Science and Engineering**

**School of Computer Science and Engineering**

**Manipal University Jaipur**

**Jaipur, Rajasthan - 303007**

Internship Project Report

On

# Machine Learning Capstone Project

Submitted to

LaunchEd

By

Anwesha Singh

anweshasingh0611@gmail.com

(B.Tech. 2023-2027)

**CONTENTS**




https://github.com/Anwesha-code/LaunchEd_CapstoneProject
anweshasingh0611@gmail.com

# 1. INTRODUCTION

**1.1 Project Overview:** Business Problem: To ensure there is no discrimination between employees, it is imperative for the Human Resources department of Company X to maintain a salary range for each employee with a similar profile. Apart from the existing salary, a considerable number of factors, such as an employee's experience and other abilities, are evaluated during interviews. Given the data related to individuals who applied to Company X, models can be built that automatically determine the salary to be offered if a prospective candidate is selected. This model seeks to minimize human judgment in salary decisions.

**1.2 Goal & Objective:** The objective of this exercise is to build a model, using historical data, that will determine the salary to be offered to an employee, minimizing manual judgment in the selection process. The approach aims to be robust and eliminate any discrimination in salary among employees with similar profiles.

Upon reading the Problem-Statement, this seems like a **supervised regression** problem, where the goal is to predict the **salary (continuous variable)** of a candidate based on features like experience, skills, etc.

## 2. EXPLORATORY DATA ANALYSIS

Step in which we analyze, understand, and summarize the characteristics of the dataset. It is the first step before applying any ML model.

| IDX | Applicant_ | Total_Expe | Total_Expe | Departmer | Role | Industry | Organizatic | Designatio | Education | Graduatior | University_ | Passing_Ye | PG_Specia | University_ | Passing_Ye | PHD_Spec | University_ | Passing_Ye | Curent_Lo | Preferred_l | Current_C | Inhand_Off | Last_Appra | No_Of_Co | Number_o | Certificatic | Internation | Expected_CTC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 22753 | 0 | 0 | NA | NA | NA | NA | NA | PG | Arts | Lucknow | 2020 | | | | | | | Guwahati | Pune | 0 | N | NA | 0 | 0 | 0 | 0 | 384551 |
| 2 | 51087 | 23 | 14 | HR | Consultant | Analytics | H | HR | Doctorate | Chemistry | Surat | 1988 | Others | Surat | 1990 | Chemistry | Mangalore | 1997 | Bangalore | Nagpur | 2702664 | Y | Key_Perfor | 2 | 4 | 0 | 0 | 3783729 |
| 3 | 38413 | 21 | 12 | Top Manag | Consultant | Training | J | | Doctorate | Zoology | Jaipur | 1990 | Zoology | Jaipur | 1992 | Zoology | Lucknow | 1999 | Ahmedaba | Jaipur | 2236661 | Y | Key_Perfor | 5 | 3 | 0 | 0 | 3131325 |
| 4 | 11501 | 15 | 8 | Banking | Financial A | Aviation | F | HR | Doctorate | Others | Bangalore | 1997 | Zoology | Bangalore | 1999 | Chemistry | Guwahati | 2005 | Kanpur | Kolkata | 2100510 | N | C | 5 | 3 | 0 | 0 | 2608833 |
| 5 | 58941 | 10 | 5 | Sales | Project Ma | Insurance | E | Medical Of | Grad | Zoology | Mumbai | 2004 | Zoology | Mumbai | 2006 | Zoology | Bangalore | 2010 | Ahmedaba | Ahmedaba | 1931644 | N | C | 2 | 3 | 0 | 0 | 2221390 |
| 6 | 30564 | 16 | 3 | Top Manag | Area Sales | Retail | G | Director | Doctorate | Others | Bangalore | 1998 | Zoology | Bangalore | 2000 | Others | Bhubanesv | 2004 | Pune | Bhubanesv | 3511167 | Y | C | 5 | 4 | 0 | 0 | 4522383 |
| 7 | 27267 | 1 | 1 | Engineerin | Team Lead | FMCG | L | Marketing I | Grad | Chemistry | Delhi | 2011 | Chemistry | Delhi | 2013 | Chemistry | Mangalore | 2019 | Delhi | Pune | 492917 | Y | B | 3 | 3 | 0 | 0 | 630933 |
| 8 | 36521 | 19 | 11 | Others | Analyst | Others | E | Manager | PG | Sociology | Delhi | 2001 | | | | | | | Surat | Bangalore | 2693459 | N | C | 6 | 1 | 1 | 0 | 3221376 |
| 9 | 11616 | 8 | 7 | Analytics/E | Others | Telecom | L | Marketing I | Doctorate | Psycholog | Mumbai | 2003 | Psycholog | Mumbai | 2005 | Psycholog | Pune | 2012 | Nagpur | Guwahati | 1630225 | N | A | 6 | 6 | 0 | 0 | 2288835 |
| 10 | 43886 | 15 | 15 | Analytics/E | CEO | Telecom | M | Product Ma | Doctorate | Chemistry | Delhi | 1998 | Chemistry | Delhi | 2000 | Chemistry | Jaipur | 2005 | Bangalore | Bhubanesv | 1761797 | Y | A | 5 | 6 | 0 | 1 | 2466515 |
| 11 | 47373 | 13 | 10 | Education | Business A | Automobile | L | Consultant | Doctorate | Mathemati | Mangalore | 2000 | Mathemati | Mangalore | 2002 | Others | Guwahati | 2007 | Bangalore | Nagpur | 2232801 | Y | Key_Perfor | 3 | 3 | 0 | 0 | 3125921 |
| 12 | 36565 | 7 | 1 | Marketing | Sales Man | FMCG | E | CA | PG | Others | Mumbai | 2004 | Chemistry | Mumbai | 2006 | Others | Surat | 2013 | Jaipur | Mangalore | 1351191 | N | B | 2 | 8 | 1 | 0 | 1756548 |
| 13 | 11739 | 10 | 10 | Others | Bio statisti | Automobile | H | Consultant | Under Grad | | | | | | | | | | Surat | Kanpur | 1446451 | N | C | 3 | 2 | 1 | 0 | 1663418 |
| 14 | 14459 | 0 | 0 | NA | NA | NA | NA | NA | PG | Engineerin | Nagpur | 2012 | Engineerin | Nagpur | 2014 | Engineerin | Kolkata | 2020 | Nagpur | Bangalore | 0 | N | NA | 0 | 6 | 0 | 0 | 639655 |
| 15 | 45111 | 12 | 9 | Banking | Bio statisti | Telecom | J | Research S | Grad | Zoology | Surat | 2002 | Others | Surat | 2004 | Others | Delhi | 2008 | Jaipur | Mumbai | 1990865 | N | C | 6 | 3 | 0 | 0 | 2289494 |
| 16 | 29106 | 20 | 15 | Healthcare | Analyst | IT | F | Medical Of | PG | Sociology | Mumbai | 2000 | Sociology | Mumbai | 2014 | | | | Kolkata | Kolkata | 2624073 | N | B | 5 | 1 | 0 | 0 | 3411294 |

2.1 Snip-it of the dataset

| COLUMN NAME | CONTENT | DATATYPE | USE |
|---|---|---|---|
| IDX | Index | ID | Ignore |
| Applicant_ID | Unique ID for each applicant | Categorical | Might be helpful |
| Total_Experience | Years of work experience | Numerical | Strongly helpful in salary prediction |
| Total_Experience_in_field_applied | Experience for the role applied | Numerical | Important |
| Department | Department to which the role belongs to | Categorical | Used to assign department |
| Role | Applied job role | Categorical | Significant impact on salary |
| Industry | Industry where applicant previously worked at | Categorical | Might be helpful |
| Organization | Previous company | Categorical | Might be helpful |
| Designation | Designation at previous job | Categorical | Decides seniority |
| Education | Education level | Categorical | Important influence |
| Graduation_Specialization | Degree subject | Categorical | Might have a minor impact |
| University_Grad | Undergraduate university | Categorical | Useful if elite universities are favoured |
| Passing_Year_Of_Graduation | Year of graduation | Numerical | Little influence |
| PG_Specialization | PG degree subject | Categorical | Might have some influence |
| University_PG | PG university | Categorical | Useful if elite universities are favoured |
| Passing_Year_Of_PG | Year PG degree was completed | Numerical | Little influence |
| PHD_Specialization | PHD subject | Categorical | |
| University_PHD | PHD university | Categorical | Useful if elite universities are favoured |
| Passing_Year_Of_PHD | Year of PHD completion | Numerical | |
| Current_Location | Where applicant is currently located | Categorical | Might have impact |
| Preferred_Location | Location where applicant wants to work | Categorical | Impacts applicant's willingness to work |
| Current_CTC | Current salary | Numerical | Greatly impacts |
| Inhand_Offer | If they have any other offers | Binary | Influences applicant's salary demand |
| Last_Appraisal_Rating | Performance rating at previous organization | Categorical | |
| No_Of_Companies_Worked | Number of companies the | Numerical | |

| | | | |
|---|---|---|---|
| | applicant has worked at | | |
| Number_Of_ Publications | Research papers of publications | Numerical | Might influence |
| Certifications | Number of certifications held | Numerical | Might help when determining skills |
| International_degree _any | If an applicant has any international degrees | Binary | Might influence |
| Expected_CTC | Target variable | Numerical | |

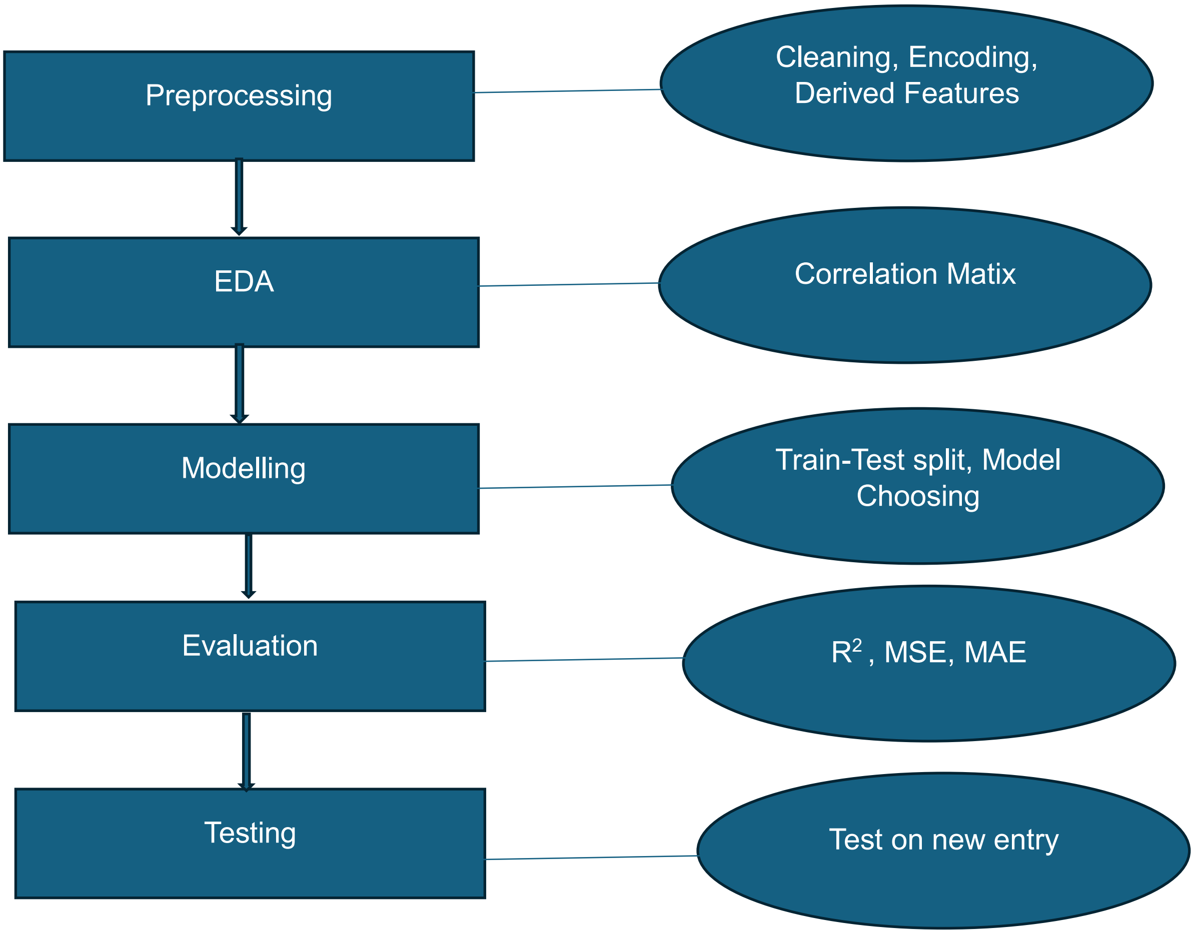
2.2 Data Analysis

### 2.3 MAJOR OBSERVATIONS IN THE DATASET

- There are many categorical features in the given dataset.
- Some features (like Organization, University_Grad) are likely to have a very high cardinality.
- Strong predictors are- Total_Experience, Current_CTC, Certifications
- There are a lot of NA values or missing values.

## 3. ROADMAP

`


Preprocessing
Cleaning, Encoding, Derived Features
EDA
Correlation Matix
Modelling
Train-Test split, Model Choosing
Evaluation
R2 , MSE, MAE
Testing
Test on new entry

## 4. UPLOADING THE EXCEL(DATASET) FILE AS PROVIDED

We were provided with the dataset with name 'exprcted_ctc'. To use a model, we must upload the dataset as the first step.

This step is the same regardless the model you use.

### ✅ UPLOADING THE EXCEL(DATASET) FILE AS PROVIDED

We were provided with the dataset with name 'exprcted_ctc'. To use a model, we must upload the dataset as the first step.

```
[ ] from google.colab import files
    uploaded = files.upload()
```

```
Choose Files  No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving expected_ctc.csv to expected_ctc.csv
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('expected_ctc.csv')
print(df.shape)
print(df.count())
```

```
(25000, 29)
IDX                                  25000
Applicant_ID                         25000
Total_Experience                     25000
Total_Experience_in_field_applied    25000
Department                           22222
Role                                 24037
Industry                             24092
Organization                         24092
Designation                          21871
Education                            25000
Graduation_Specialization            18820
University_Grad                      18820
Passing_Year_Of_Graduation           18820
PG_Specialization                    17308
University_PG                        17308
Passing_Year_Of_PG                   17308
PHD_Specialization                   13119
University_PHD                       13119
Passing_Year_Of_PHD                  13119
Curent_Location                      25000
Preferred_location                   25000
Current_CTC                          25000
Inhand_Offer                         25000
Last_Appraisal_Rating                24092
No_Of_Companies_worked               25000
Number_of_Publications               25000
Certifications                       25000
International_degree_any             25000
Expected_CTC                         25000
dtype: int64
```


https://github.com/Anwesha-code/LaunchEd_CapstoneProject
anweshasingh0611@gmail.com

## 5. CLEANING THE DATASET

Fixing, or removing incorrect, incomplete, or irrelevant data before training the model.

- Replace null values or empty strings
- Create derived features as required
- Dropping any unnecessary columns
- Filling any missing values
- Encode categorical or binary columns (there are many categorical features.)
- One hot encode other categorical variables

```
import numpy as np
import pandas as pd

df.replace(['NA', ''], np.nan, inplace=True)                #Replace NA and empty strings

if 'Passing_Year_Of_Graduation' in df.columns:                #Create derived features
    df['Years_Since_Grad'] = 2025 - df['Passing_Year_Of_Graduation'].astype(float)

if 'Passing_Year_Of_PG' in df.columns:
    df['Years_Since_PG'] = 2025 - df['Passing_Year_Of_PG'].astype(float)

columns_to_drop = [                              #Drop unecessary columns
'IDX', 'Applicant_ID', 'Organization',
'University_Grad', 'University_PG', 'University_PHD',
'PHD_Specialization', 'Passing_Year_Of_PHD',
'Passing_Year_Of_Graduation', 'Passing_Year_Of_PG',
'Number_of_Publications']
existing_cols = [col for col in columns_to_drop if col in df.columns]
df.drop(columns=existing_cols, inplace=True)

#Filling missing values
df['Total_Experience'] = df['Total_Experience'].fillna(df['Total_Experience'].median())
df['Certifications'] = df['Certifications'].fillna(0)
df['Role'] = df['Role'].fillna('Unknown')
df['Education'] = df['Education'].fillna('Grad')

df['Inhand_Offer'] = df['Inhand_Offer'].map({'Y': 1, 'N': 0})          #Encode categorical or binary columns
df['International_degree_any'] = df['International_degree_any'].astype(int)
rating_map = {'A': 3, 'B': 2, 'C': 1, 'Key_Performer': 4}
df['Last_Appraisal_Rating'] = df['Last_Appraisal_Rating'].map(rating_map)

df = pd.get_dummies(df, columns=[            #One hot encode any other categorical variables
    'Department', 'Role', 'Industry', 'Designation',
    'Education', 'Graduation_Specialization', 'PG_Specialization'], drop_first=True)

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

df.drop(columns=['Curent_Location', 'Preferred_location'], inplace=True)        #Drop unneeded location columns

df['Last_Appraisal_Rating'] = df['Last_Appraisal_Rating'].fillna(df['Last_Appraisal_Rating'].median())    # Fill missing values in ratings
df['Years_Since_Grad'] = df['Years_Since_Grad'].fillna(df['Years_Since_Grad'].median())
df['Years_Since_PG'] = df['Years_Since_PG'].fillna(df['Years_Since_PG'].median())
df.rename(columns={'Curent_Location': 'Current_Location'}, inplace=True)

low_freq_cols = [col for col in df.columns if df[col].dtype == 'bool' and df[col].sum() < (0.005 * len(df))]        # Drop very sparse one-hot encoded features
df.drop(columns=low_freq_cols, inplace=True)

print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
print(df.info())
print(df.isnull().sum())
print(df.head())
```


https://github.com/Anwesha-code/LaunchEd_CapstoneProject
anweshasingh0611@gmail.com

```
Total_Experience                   0
Total_Experience_in_field_applied  0
Curent_Location                    0
Preferred_location                 0
Current_CTC                        0
                                  ..
PG_Specialization_Others           0
PG_Specialization_Psychology       0
PG_Specialization_Sociology        0
PG_Specialization_Statistics       0
PG_Specialization_Zoology          0
Length: 98, dtype: int64
0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 90 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Total_Experience                    25000 non-null  int64
 1   Total_Experience_in_field_applied   25000 non-null  int64
 2   Current_CTC                         25000 non-null  int64
 3   Inhand_Offer                        25000 non-null  int64
 4   Last_Appraisal_Rating               25000 non-null  float64
 5   No_Of_Companies_worked              25000 non-null  int64
 6   Certifications                      25000 non-null  int64
 7   International_degree_any            25000 non-null  int64
 8   Expected_CTC                        25000 non-null  int64
 9   Years_Since_Grad                    25000 non-null  float64
 10  Years_Since_PG                      25000 non-null  float64
 11  Department_Analytics/BI             25000 non-null  bool
 12  Department_Banking                  25000 non-null  bool
 13  Department_Education                25000 non-null  bool
 84  PG_Specialization_Mathematics       25000 non-null  bool
 85  PG_Specialization_Others            25000 non-null  bool
 86  PG_Specialization_Psychology        25000 non-null  bool
 87  PG_Specialization_Sociology         25000 non-null  bool
 88  PG_Specialization_Statistics        25000 non-null  bool
 89  PG_Specialization_Zoology           25000 non-null  bool
dtypes: bool(79), float64(3), int64(8)
memory usage: 4.0 MB
None
Total_Experience                   0
Total_Experience_in_field_applied  0
Current_CTC                        0
Inhand_Offer                       0
Last_Appraisal_Rating              0
                                  ..
PG_Specialization_Others           0
PG_Specialization_Psychology       0
PG_Specialization_Sociology        0
PG_Specialization_Statistics       0
PG_Specialization_Zoology          0
Length: 90, dtype: int64
   Total_Experience  Total_Experience_in_field_applied  Current_CTC  \
0                 0                                  0            0
1                23                                 14      2702664
2                21                                 12      2236661
3                15                                  8      2100510
4                10                                  5      1931644

   Inhand_Offer  Last_Appraisal_Rating  No_Of_Companies_worked  \
0             0                    2.0                       0
1             1                    4.0                       2
2             1                    4.0                       5
```


anweshasingh0611@gmail.com

```
   PG_Specialization_Economics  PG_Specialization_Engineering  \
0                        False                          False
1                        False                          False
2                        False                          False
3                        False                          False
4                        False                          False

   PG_Specialization_Mathematics  PG_Specialization_Others  \
0                          False                     False
1                          False                      True
2                          False                     False
3                          False                     False
4                          False                     False

   PG_Specialization_Psychology  PG_Specialization_Sociology  \
0                         False                        False
1                         False                        False
2                         False                        False
3                         False                        False
4                         False                        False

   PG_Specialization_Statistics  PG_Specialization_Zoology
0                         False                     False
1                         False                     False
2                         False                      True
3                         False                      True
4                         False                      True

[5 rows x 90 columns]
```


anweshasingh0611@gmail.com

## 6. TRAIN-TEST SPLIT AND TRAINING THE MODEL

Dividing the dataset into two parts:

- **Training set-** I used 80% of dataset to train the model.
- **Testing set-** I used 20% of dataset to do the testing of the model.

It helps when checking if the model generalizes well and if it avoids overfitting.

**CHOOSING THE MODEL**

Based on the Exploratory Data Analysis, this problem comes out to be a **regression** problem.

I will use three models on the dataset and compare each one's accuracy and give the output based on the most accurate one.

I'll be using the following three models:

- **Linear Regression-** A supervised ML model used when predicting continuous value with the help of the linear relationship between input features and target.
  It tries to find the best fitting straight line which helps minimize the difference between actual and predicted values.
- **Random Forest Regressor-** A supervised ML model which uses an ensemble of decision trees to predict continuous values. It picks out random subsets of data from the dataset and takes the average of the predictions.
- **Gradient Boosting Regressor-** A supervised ML model that makes an ensemble of weaker models in a sequential manner. Each model in the ensemble rectifies the errors of the previous model.

## 6.1 Linear Regression

### ✅ TRAIN-TEST SPLIT AND TRAINING THE MODEL

Dividing the dataset into two parts:

- **Training set-** I used 80% of Dataset to train the model.
- **Testing set-** I used 20% of Dataset to do the testing of the model.

```
X = df.drop(columns=['Expected_CTC'])
y = df['Expected_CTC']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

**HOW TO CHOOSE THE MODEL**

- In the end, we are predicting a numeric value, i.e. Expected_CTC.
- Many features like experience, education, domain, Current_CTC, etc. are to be taken in consideration.

Ultimately this is a **regression** problem, as our output (Expected CTC) is a continuous value.

---

Models that I think can be used are:

- Linear Regression
- Random Forest Regressor
- Gradient Boosting Regressor

---

I will use all these three models, and will check each of their accuracy.

Here I am using **Linear Regression Model**.

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

```
LinearRegression
LinearRegression()
```

## 6.2 Random Forest Regressor

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

X = df.drop(columns=['Expected_CTC'])
y = df['Expected_CTC']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)    # Train-test split

scaler = StandardScaler()              #Scaling the data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

**HOW TO CHOOSE THE MODEL**

- In the end, we are predicting a numeric value, i.e. Expected_CTC.
- Many features like experience, education, domain, Current_CTC, etc. are to be taken in consideration. Ultimately this is a **regression** problem, as our output (Expected CTC) is a continuous value.

Models that I think can be used are:

- Linear Regression
- Random Forest Regressor
- Gradient Boosting Regressor

I will use all these three models, and will check each of their accuracy.

```
[ ] rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(X_train_scaled, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

## 6.3 Gradient Boosting Regressor

```
from sklearn.model_selection import train_test_split

X = df.drop(columns=['Expected_CTC'])
y = df['Expected_CTC']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)      #spliting the data
```

```
[ ] from sklearn.preprocessing import StandardScaler

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
```

**HOW TO CHOOSE THE MODEL**

- In the end, we are predicting a numeric value, i.e. Expected_CTC.
- Many features like experience, education, domain, Current_CTC, etc. are to be taken in consideration.

Ultimately this is a regression problem, as our output (Expected CTC) is a continuous value.

Models that I think can be used are:

- Linear Regression
- Random Forest Regressor
- Gradient Boosting Regressor

I will use all these three models, and will check each of their accuracy.

Here I am using Gradient Boosting Regressor Model.

```
[ ] from sklearn.ensemble import GradientBoostingRegressor
    from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
    import numpy as np

    gbr = GradientBoostingRegressor(random_state=42)      #initializing and train model
    gbr.fit(X_train_scaled, y_train)

    y_pred = gbr.predict(X_test_scaled)       #predicting
```

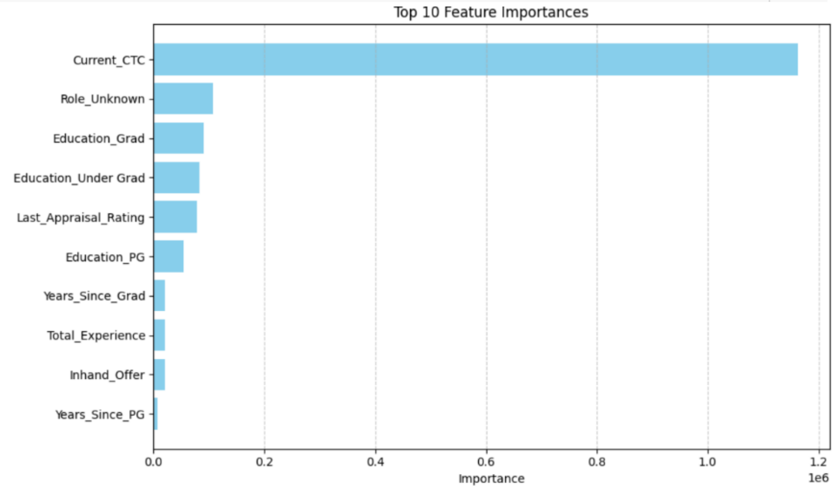
## 7. FEATURE IMPORTANCE

Helps us determine which input columns have the biggest impact on model's predictions.

Useful in selecting the features for predictions. It also helps improve model's performance and accuracy.

### 7.1 Linear Regression

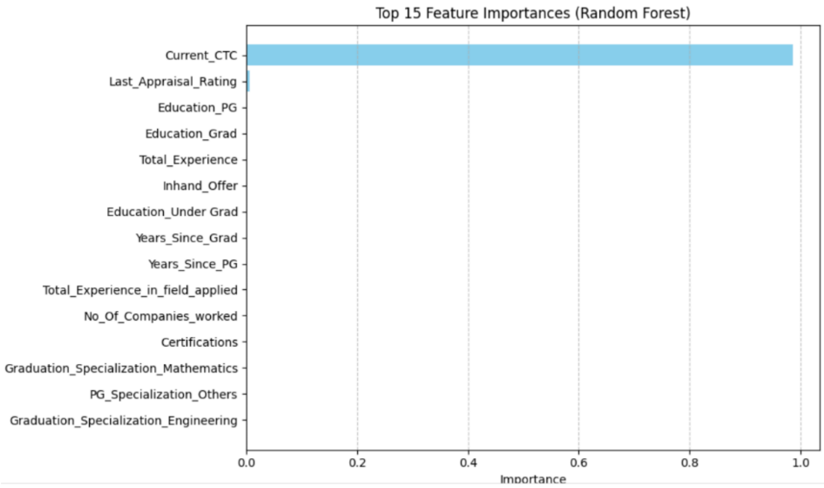
```python
import matplotlib.pyplot as plt

# Plot the top 10 features
top_features = importance_df.head(10)

plt.figure(figsize=(10, 6))
plt.barh(top_features['Feature'], top_features['Importance'], color='skyblue')
plt.xlabel('Importance')
plt.title('Top 10 Feature Importances')
plt.gca().invert_yaxis()  # Highest importance on top
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```




### 7.2 Random Forest Regressor

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

importances = rf.feature_importances_
indices = np.argsort(importances)[-15:]

feature_names = [X.columns[i] for i in indices]
importance_values = importances[indices]
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importance_values}).sort_values(by='Importance', ascending=False)

print("Top 15 Feature Importances:")
print(importance_df.to_string(index=False))

plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='skyblue')
plt.xlabel("Importance")
plt.title("Top 15 Feature Importances (Random Forest)")
plt.gca().invert_yaxis()
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Top 15 Feature Importances:
                                  Feature  Importance
                              Current_CTC    0.985505
                     Last_Appraisal_Rating   0.005523
                             Education_PG    0.001482
                           Education_Grad    0.001241
                         Total_Experience    0.000917
                             Inhand_Offer    0.000786
                     Education_Under Grad    0.000784
                          Years_Since_Grad   0.000499
                            Years_Since_PG   0.000459
            Total_Experience_in_field_applied 0.000292
                      No_Of_Companies_worked  0.000275
                            Certifications   0.000152
      Graduation_Specialization_Mathematics   0.000079
                  PG_Specialization_Others    0.000074
      Graduation_Specialization_Engineering   0.000074
```

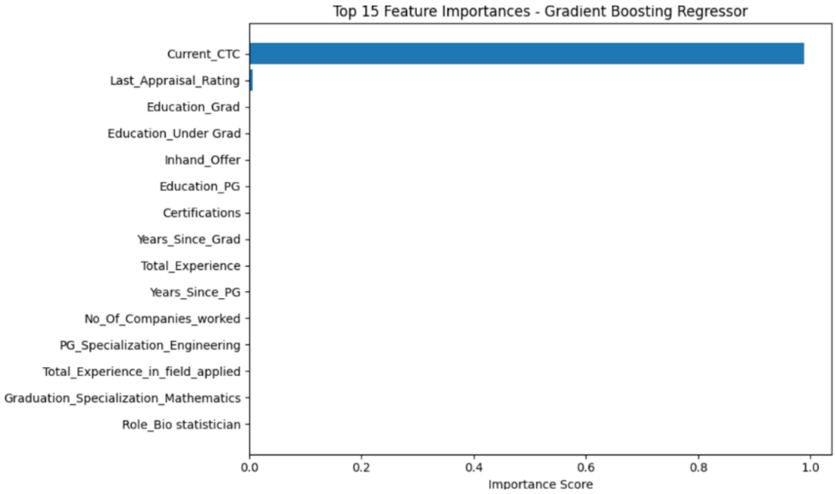



## 7.3 Gradient Boosting Regressor

```
import matplotlib.pyplot as plt

top_features = importance_df.head(10)

plt.figure(figsize=(10, 6))
plt.barh(top_features['Feature'], top_features['Importance'], color='skyblue')
plt.xlabel('Importance')
plt.title('Top 10 Feature Importances')
plt.gca().invert_yaxis()
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
[10] import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

importances = gbr.feature_importances_

feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

print(feature_importance_df.head(15))

plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'][:15][::-1], feature_importance_df['Importance'][:15][::-1])
plt.title("Top 15 Feature Importances - Gradient Boosting Regressor")
plt.xlabel("Importance Score")
plt.tight_layout()
plt.show()
```

```
                                       Feature  Importance
2                                  Current_CTC    0.987985
4                          Last_Appraisal_Rating    0.005519
66                              Education_Grad    0.002015
68                        Education_Under Grad    0.001330
3                                 Inhand_Offer    0.001094
67                                Education_PG    0.000853
6                               Certifications    0.000452
8                             Years_Since_Grad    0.000283
0                             Total_Experience    0.000171
9                               Years_Since_PG    0.000165
5                       No_Of_Companies_worked    0.000097
82                  PG_Specialization_Engineering    0.000011
1              Total_Experience_in_field_applied    0.000010
73   Graduation_Specialization_Mathematics    0.000006
23                         Role_Bio statistician    0.000003
```






https://github.com/Anwesha-code/LaunchEd_CapstoneProject
anweshasingh0611@gmail.com

## 8. MODEL EVALUATION

Check how well the model performs after training on unseen test data. It helps measure the accuracy and errors.

We often use following metrics to evaluate a model.

**MSE:**

Measures the average squared difference between actual and predicted values.

Useful for understanding how far predictions are from actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- $y_i$ = actual value
- $\hat{y}_i$ = predicted value
- $n$ = number of samples

**Lower MSE = better model**

**R² Score**:

Coefficient of Determination. Used in regression to measure how well a model explains variances in the target column.

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

Where:

- $\text{SS}_{\text{res}}$ = sum of squared residuals (errors)
- $\text{SS}_{\text{tot}}$ = total sum of squares (variance in actual data)

Ranges between 0 to 1 (or negative if very poor).

Tells how good your model is compared to a baseline (mean) model.

$R^2 = 1$ means perfect prediction, $R^2 = 0$ means predictions are no better than guessing the mean.

**MAE:**

Used in regression tasks to measure average absolute difference between the actual values and the predicted values.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where:

- $y_i$ = actual value
- $\hat{y}_i$ = predicted value
- $n$ = number of samples

### 8.1 Linear Regressor Model

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"R²: {r2:.4f}")
print(f"MSE: {mse:,.4f}")
```

```
R²: 0.9926
MSE: 10,049,990,051.6156
```

### 8.2 Random Forest Regressor

```
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np

y_pred = rf.predict(X_test_scaled)

print("R² Score:", r2_score(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
```

```
R² Score: 0.9958806849100306
MSE: 5571275972.783264
RMSE: 74640.9805186351
```

### 8.3 Gradient Boosting Regressor

```
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np

print("R² Score:", r2_score(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

```
R² Score: 0.9954361344875179
MSE: 6172519877.059157
```

## 9. COMPARING THE ACCURACY OF EACH MODEL

- Model evaluation output for **'Linear Regression'** model:
  **$R^2$:** 0.9926
  **MSE:** 10,049,990,051.6156
- Model evaluation output for **'Random Forest Regressor'** model:
  **$R^2$:** 0.9958806849100306
  **MSE:** 5571275972.783264
- Model evaluation output for **'Gradient Boosting Regressor'** model:
  **$R^2$:** 0.9954361344875179
  **MSE:** 6172519877.059157

Comparing these models based on these outputs:

**Random Forest Regressor** has the highest $R^2$ score, which is it experiences about **99.59%** variances in salaries.

It also has **the lowest Mean Squared Error**, which means that it has the lowest average prediction error among the others.

**Hence, the outputs from Random Forest Regressor model are the most accurate and dependable.**

## 10.TESTING THE MODEL

I created a dictionary with a random candidate's entries to test the model.

- **Linear Regressor**
  Expected Salary (CTC): ₹1,169,836.46

I created a dictionary with a random candidate's entries to test the model.

```
candidate_dict = dict.fromkeys(X.columns, 0)          # starting with all features = 0

candidate_dict.update({'Total_Experience': 6,
'Total_Experience_in_field_applied': 5,
'Current_CTC': 900000,
'Inhand_Offer': 0,
'Last_Appraisal_Rating': 3,
'No_Of_Companies_worked': 2,
'Certifications': 1,
'International_degree_any': 0,
'Years_Since_Grad': 4,
'Years_Since_PG': 2,
'Role_Data scientist': 1,              #set the correct one-hot fields = 1
'Department_IT-Software': 1,
'Education_PG': 1,
'PG_Specialization_Engineering': 1})

new_candidate_df = pd.DataFrame([candidate_dict])          #creating a dataframe from the dictionary created above

new_candidate_scaled = scaler.transform(new_candidate_df)          #scaling the input

predicted_salary = model.predict(new_candidate_scaled)[0]          #predicting the salary
print(f"Suggested salary: ₹{predicted_salary:,.2f}")
```

```
Suggested salary: ₹1,169,836.46
```

- **Random Forest Regressor**
  Expected Salary (CTC): ₹1,214,645.26 (Most likely, most accurate)

```
import pandas as pd

candidate_dict = {
    'Total_Experience': 6,
    'Total_Experience_in_field_applied': 5,
    'Current_CTC': 900000,
    'Inhand_Offer': 0,
    'Last_Appraisal_Rating': 3,
    'No_Of_Companies_worked': 2,
    'Certifications': 1,
    'International_degree_any': 0,
    'Years_Since_Grad': 4,
    'Years_Since_PG': 2,
    'Role_Data scientist': 1,              #set the correct one-hot fields = 1
    'Department_IT-Software': 1,
    'Education_PG': 1,
    'PG_Specialization_Engineering': 1}

candidate_df = pd.DataFrame([candidate_dict])          #creating a dataframe from the dictionary created above

candidate_df = candidate_df.reindex(columns=X.columns, fill_value=0)

candidate_scaled = scaler.transform(candidate_df)          #scaling the input

predicted_salary = rf.predict(candidate_scaled)[0]

print(f"Expected Salary (CTC): ₹{predicted_salary:,.2f}")
```

```
Expected Salary (CTC): ₹1,214,645.26
```

- **Gradient Booster Regressor**
  Expected Salary (CTC): ₹1,085,973.99

```python
candidate_dict = {
    'Total_Experience': 6,
    'Total_Experience_in_field_applied': 5,
    'Current_CTC': 900000,
    'Inhand_Offer': 0,
    'Last_Appraisal_Rating': 3,
    'No_Of_Companies_worked': 2,
    'Certifications': 1,
    'International_degree_any': 0,
    'Years_Since_Grad': 4,
    'Years_Since_PG': 2,
    'Role_Data scientist': 1,        #set the correct one-hot fields = 1
    'Department_IT-Software': 1,
    'Education_PG': 1,
    'PG_Specialization_Engineering': 1}


missing_cols = set(X.columns) - set(candidate_dict.keys())
for col in missing_cols:
    candidate_dict[col] = 0

candidate_df = pd.DataFrame([candidate_dict])[X.columns]
candidate_scaled = scaler.transform(candidate_df)

predicted_salary = gbr.predict(candidate_scaled)[0]
print(f"Expected Salary (CTC): ₹{predicted_salary:,.2f}")
```

```
Expected Salary (CTC): ₹1,085,973.99
```


https://github.com/Anwesha-code/LaunchEd_CapstoneProject
anweshasingh0611@gmail.com

## 11. CONCLUSION

Throughout this project, the goal was to make a ML model predicting expected salary for applicants applying for a job at some company X, while minimizing manual judgement and eliminating any discrimination in salary among applicants with similar qualifications.

To achieve this, I implemented three regression models:

- Linear Regression
- Random Forest Regressor
- Gradient Boosting Regressor

Dataset was cleaned and preprocessed thoroughly and then each model was trained.

Each model was evaluated on metrics such as $R^2$ score and Mean Squared Error (MSE). The **Random Forest Regressor model** showed the best and most accurate results and performed the best, with the $R^2$ score of 0.9959 and lowest MSE, which signifies that it was the most accurate.

Hence, if we predict salaries using Random Forest regressor we can significantly ensure reduced discrimination and accuracy. This model can assist the company to get the job done while ensuring transparency.

Link to my code repository on github:

https://github.com/Anwesha-code/LaunchEd_CapstoneProject