

# SALARY PREDICTION MODEL REPORT

**Team Member :** *Dipali Gantait , Sudipta Singha , Riyajul Saha*

**Instructor :** *Sourav Goswami (Ardent Computech PVT LTD )*

**DATE :** *18/06/2025*

## ➤ Abstract :

This project focuses on building a machine learning model to predict salaries using Linear Regression. The goal is to estimate an individual's salary based on key job-related features such as job title, experience, education level . The dataset was carefully preprocessed, including encoding categorical data and handling missing values, to ensure model accuracy. Linear Regression was chosen as the core algorithm due to its simplicity and effectiveness in modeling linear relationships. After training and evaluating the model, it achieved an  $R^2$  score greater than 0.85, indicating a strong correlation between the input features and salary outcomes. This Salary Prediction Model can be useful for job seekers and HR professionals to estimate fair and data-driven salary expectations.

## ➤ Problem Statement :

In today's competitive job market, determining a fair and accurate salary for a given job role is a complex task. Salaries can vary widely depending on factors such as job title, level of experience, and educational background. Both job seekers and employers often struggle to make data-driven decisions regarding salary expectations and offers. This project aims to address this issue by developing a machine learning model that can predict an individual's salary based on key job-related attributes. By analyzing historical salary data using Linear Regression, the model provides reliable salary estimations, reducing guesswork and supporting fair compensation practices.

## ➤ Objectives :

1. To collect and understand the dataset containing features such as job title, experience, and education level related to salary.
2. To preprocess the dataset by handling missing values and encoding categorical variables for effective model training.
3. To build and train a machine learning model using Linear Regression to learn the relationship between input features and salary.
4. To evaluate the model using performance metrics such as  $R^2$  score to ensure prediction accuracy.
5. To save the trained model using the .pkl (Pickle) format for easy reuse and deployment.

## ➤ Tools & Technologies :

1. Python – Programming language used for data processing and model development.
2. Pandas – For data manipulation, cleaning, and analysis.
3. NumPy – For numerical computations and array operations.
4. Scikit-learn (sklearn) – For implementing Linear Regression, preprocessing (encoding), and evaluating the model.
5. Google Colab – Cloud-based environment used to write, execute, and test the Python code.

6. Pickle – To save the trained model in .pkl format for later use.

## ➤ Dataset Description :

The dataset used in this project initially contained 375 records and 6 columns, representing individuals with various job-related attributes and their corresponding salaries. After data cleaning, Age and Gender columns were removed to simplify the model and focus on the most relevant features.

1. Education Level (object) – The highest education qualification (e.g., Bachelor's, Master's, PhD).
2. Job Title (object) – The current job title of the individual.
3. Years of Experience (float) – The number of years the person has worked in their field.
4. Salary (float) – The annual salary of the individual (target variable).

## ➤ import necessary libraries :

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

## ➤ Dataset loading and exploration

we already uploaded our Dataset into mydrive-Dataset folder. So before loading Dataset in pandas Dataframe we need to connect google colab with mydrive.

```
from google.colab import drive
drive.mount('/content/drive')
```

➡ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive")

Now we load our Salary.csv file as Dataframe using pandas

```
df=pd.read_csv('/content/drive/My Drive/Dataset/Salary.csv')
```

our dataset loading completed now we see first 5 rows of Dataframe

```
df.head()
```



	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0

Display basic information about the Dataframe

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 375 entries, 0 to 374
Data columns (total 6 columns):
#   Column              Non-Null Count  Dtype
#   ...
```

```

---  -----
0   Age                373 non-null    float64
1   Gender              373 non-null    object
2   Education Level     373 non-null    object
3   Job Title           373 non-null    object
4   Years of Experience  373 non-null    float64
5   Salary              373 non-null    float64
dtypes: float64(3), object(3)
memory usage: 17.7+ KB

```

df.shape

➦ (375, 6)

df.describe()

➦

	Age	Years of Experience	Salary
<b>count</b>	373.000000	373.000000	373.000000
<b>mean</b>	37.431635	10.030831	100577.345845
<b>std</b>	7.069073	6.557007	48240.013482
<b>min</b>	23.000000	0.000000	350.000000
<b>25%</b>	31.000000	4.000000	55000.000000
<b>50%</b>	36.000000	9.000000	95000.000000
<b>75%</b>	44.000000	15.000000	140000.000000
<b>max</b>	53.000000	25.000000	250000.000000

## ➤ Data Cleaning and Preprocessing:

From above information there has Age and Gender column, that are not very important for our model. So we drop them.

```
df.drop(["Age", "Gender"], axis=1, inplace=True)
```

df.head()

➦

	Education Level	Job Title	Years of Experience	Salary
<b>0</b>	Bachelor's	Software Engineer	5.0	90000.0
<b>1</b>	Master's	Data Analyst	3.0	65000.0
<b>2</b>	PhD	Senior Manager	15.0	150000.0
<b>3</b>	Bachelor's	Sales Associate	7.0	60000.0
<b>4</b>	Master's	Director	20.0	200000.0

Now we will check if there has null value or not.

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

```
↵↵
```

	0
Education Level	2
Job Title	2
Years of Experience	2
Salary	2

**dtype:** int64

In Dataset there are six cell has null value, those are very small value with respect to Dataframe , so we will drop corresponding row.

```
df.dropna(how="any",inplace=True)
```

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

```
↵↵
```

	0
Education Level	0
Job Title	0
Years of Experience	0
Salary	0

**dtype:** int64

```
df.shape
```

```
↵↵ (373, 4)
```

Check for duplicates and drop them .

```
df.duplicated().sum()
```

```
↵↵ np.int64(61)
```

```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

```
↵↵ np.int64(0)
```

Now we will see unique values in categorical columns that help to find out if there has unsatisfied value in columns.

```
df["Education Level"].unique()
```

```
→ array(['Bachelor's', 'Master's', 'PhD'], dtype=object)
```

```
df["Job Title"].unique()
```

```
→ array(['Software Engineer', 'Data Analyst', 'Senior Manager',  
        'Sales Associate', 'Director', 'Marketing Analyst',  
        'Product Manager', 'Sales Manager', 'Marketing Coordinator',  
        'Senior Scientist', 'Software Developer', 'HR Manager',  
        'Financial Analyst', 'Project Manager', 'Customer Service Rep',  
        'Operations Manager', 'Marketing Manager', 'Senior Engineer',  
        'Data Entry Clerk', 'Sales Director', 'Business Analyst',  
        'VP of Operations', 'IT Support', 'Recruiter', 'Financial Manager',  
        'Social Media Specialist', 'Software Manager', 'Junior Developer',  
        'Senior Consultant', 'Product Designer', 'CEO', 'Accountant',  
        'Data Scientist', 'Marketing Specialist', 'Technical Writer',  
        'HR Generalist', 'Project Engineer', 'Customer Success Rep',  
        'Sales Executive', 'UX Designer', 'Operations Director',  
        'Network Engineer', 'Administrative Assistant',  
        'Strategy Consultant', 'Copywriter', 'Account Manager',  
        'Director of Marketing', 'Help Desk Analyst',  
        'Customer Service Manager', 'Business Intelligence Analyst',  
        'Event Coordinator', 'VP of Finance', 'Graphic Designer',  
        'UX Researcher', 'Social Media Manager', 'Director of Operations',  
        'Senior Data Scientist', 'Junior Accountant',  
        'Digital Marketing Manager', 'IT Manager',  
        'Customer Service Representative', 'Business Development Manager',  
        'Senior Financial Analyst', 'Web Developer', 'Research Director',  
        'Technical Support Specialist', 'Creative Director',  
        'Senior Software Engineer', 'Human Resources Director',  
        'Content Marketing Manager', 'Technical Recruiter',  
        'Sales Representative', 'Chief Technology Officer',  
        'Junior Designer', 'Financial Advisor', 'Junior Account Manager',  
        'Senior Project Manager', 'Principal Scientist',  
        'Supply Chain Manager', 'Senior Marketing Manager',  
        'Training Specialist', 'Research Scientist',  
        'Junior Software Developer', 'Public Relations Manager',  
        'Operations Analyst', 'Product Marketing Manager',  
        'Senior HR Manager', 'Junior Web Developer',  
        'Senior Project Coordinator', 'Chief Data Officer',  
        'Digital Content Producer', 'IT Support Specialist',  
        'Senior Marketing Analyst', 'Customer Success Manager',  
        'Senior Graphic Designer', 'Software Project Manager',  
        'Supply Chain Analyst', 'Senior Business Analyst',  
        'Junior Marketing Analyst', 'Office Manager', 'Principal Engineer',  
        'Junior HR Generalist', 'Senior Product Manager',  
        'Junior Operations Analyst', 'Senior HR Generalist',  
        'Sales Operations Manager', 'Senior Software Developer',  
        'Junior Web Designer', 'Senior Training Specialist',  
        'Senior Research Scientist', 'Junior Sales Representative',  
        'Junior Marketing Manager', 'Junior Data Analyst',  
        'Senior Product Marketing Manager', 'Junior Business Analyst',  
        'Senior Sales Manager', 'Junior Marketing Specialist',  
        'Junior Project Manager', 'Senior Accountant', 'Director of Sales',  
        'Junior Recruiter', 'Senior Business Development Manager',  
        'Senior Product Designer', 'Junior Customer Support Specialist',  
        'Senior IT Support Specialist', 'Junior Financial Analyst',  
        'Senior Operations Manager', 'Director of Human Resources',  
        'Junior Software Engineer', 'Senior Sales Representative',
```

```
'Director of Product Management', 'Junior Copywriter',  
'Senior Marketing Coordinator', 'Senior Human Resources Manager',  
'Junior Business Development Associate', 'Senior Account Manager',  
'Senior Researcher', 'Junior HR Coordinator',
```

According to above codes we have seen there are not any unsatisfied value in categorical columns or all null value we cleaned now our Dataset is ready to preprocess.

in above `x_train.head()` we are seeing that among Job Title and Education Labels columns there are multiple job name, to identify users input job name we convert those in lower case.

```
df["Job Title"]=df["Job Title"].str.lower()  
df["Education Level"]=df["Education Level"].str.lower()
```

```
df["Education Level"].unique()
```

```
↔ array(['bachelor's', 'master's', 'phd'], dtype=object)
```

At first we will split our Dataset into training and testing form.

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

```
x=df.drop("Salary",axis=1)
```

```
y=df["Salary"]
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

```
x_train.shape
```

```
↔ (249, 3)
```

```
x_train_copy=x_train.copy()
```

As we see before there are categorical data in two columns,so we transform categorical data into numerical data using encoding.

```
from sklearn.preprocessing import OrdinalEncoder,OneHotEncoder
```

```
ord=OrdinalEncoder(categories=[["bachelor's", "master's", 'phd']])
```

```
x_train["Education Level"]=ord.fit_transform(x_train[["Education Level"]])
```

```
x_test["Education Level"]=ord.transform(x_test[["Education Level"]])
```

```
x_test.head()
```



	Education Level	Job Title	Years of Experience
232	1.0	junior research scientist	1.5
9	2.0	senior scientist	10.0
57	2.0	senior engineer	17.0
60	1.0	director of operations	23.0
25	0.0	social media specialist	3.0

```
x_train.head()
```



	Education Level	Job Title	Years of Experience
111	0.0	software project manager	9.0
212	0.0	junior business development associate	2.0
145	0.0	senior business analyst	8.0
206	0.0	junior hr generalist	4.0
78	1.0	human resources director	20.0

```
x_train.isnull().sum()
```



	0
Education Level	0
Job Title	0
Years of Experience	0

**dtype:** int64

```
ohe=OneHotEncoder(handle_unknown="ignore",sparse_output=False)
```



```
x_train_encode=ohe.fit_transform(x_train[["Job Title"]])
```

```
x_test_encode=ohe.transform(x_test[["Job Title"]])
```

```
x_train_encode
```

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

```
x_test_encode
```

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

```
encode_df=pd.DataFrame(x_train_encode,columns=ohe.get_feature_names_out(["Job Title"]))
```

```
encode_df_test=pd.DataFrame(x_test_encode,columns=ohe.get_feature_names_out(["Job Title"]))
```

```
encode_df.head()
```

⇒

	Job Title_accountant	Job Title_administrative assistant	Job Title_business analyst	Job Title_business development manager	Job Title_business intelligence analyst	Job Title_business operations analyst
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 155 columns

```
encode_df.drop("Job Title_accountant",axis=1,inplace=True)
```

```
encode_df_test.drop("Job Title_accountant",axis=1,inplace=True)
```

```
encode_df.head()
```



	Job Title_administrative assistant	Job Title_business analyst	Job Title_business development manager	Job Title_business intelligence analyst	Job Title_ceo	Job Title_chief data officer
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 154 columns

```
x_train=pd.concat([x_train.drop("Job Title",axis=1).reset_index(drop=True),encode_df],axis=1)
```

```
x_test=pd.concat([x_test.drop("Job Title",axis=1).reset_index(drop=True),encode_df_test],axis=1)
```

```
x_train.head()
```



	Education Level	Years of Experience	Job Title_administrative assistant	Job Title_business analyst	Job Title_business development manager	Job Title_business intelligence analyst
0	0.0	9.0	0.0	0.0	0.0	0.0
1	0.0	2.0	0.0	0.0	0.0	0.0
2	0.0	8.0	0.0	0.0	0.0	0.0
3	0.0	4.0	0.0	0.0	0.0	0.0
4	1.0	20.0	0.0	0.0	0.0	0.0

5 rows × 156 columns


```
x_test.head()
```



	Education Level	Years of Experience	Job Title_administrative assistant	Job Title_business analyst	Job Title_business development manager	Job Title_business intelligence analyst
0	1.0	1.5	0.0	0.0	0.0	0.0
1	2.0	10.0	0.0	0.0	0.0	0.0
2	2.0	17.0	0.0	0.0	0.0	0.0
3	1.0	23.0	0.0	0.0	0.0	0.0
4	0.0	3.0	0.0	0.0	0.0	0.0

5 rows × 156 columns

```
x_train.isnull().sum()
```



	0
<hr/>	
Education Level	0
Years of Experience	0
Job Title_administrative assistant	0
Job Title_business analyst	0
Job Title_business development manager	0
...	...
Job Title_ux designer	0
Job Title_ux researcher	0
Job Title_vp of finance	0
Job Title_vp of operations	0
Job Title_web developer	0

156 rows × 1 columns

**dtype:** int64

As we reset index of x\_train during concatenation, so we need to reset index of y\_train and drop previous index.

```
y_train = y_train.reset_index()

y_test = y_test.reset_index(drop=True)

y_train
```



	index	Salary
0	111	95000.0
1	212	40000.0
2	145	90000.0
3	206	50000.0
4	78	180000.0
...	...	...
244	190	95000.0
245	71	70000.0
246	106	50000.0
247	292	40000.0
248	102	150000.0

249 rows × 2 columns

y\_test




	Salary
0	50000.0
1	110000.0
2	140000.0
3	170000.0
4	45000.0
...	...
58	110000.0
59	100000.0
60	190000.0
61	40000.0
62	35000.0

63 rows × 1 columns

**dtype:** float64

```
y_train.drop("index",axis=1,inplace=True)
```

y\_train



	Salary
0	95000.0
1	40000.0
2	90000.0
3	50000.0
4	180000.0
...	...
244	95000.0
245	70000.0
246	50000.0
247	40000.0
248	150000.0

249 rows × 1 columns


## ➤ Model Training:

we will apply LinearRegression Algorithm to train our model and findout  $R^2$  score.If  $R^2$  score < 0.85, we will apply SVR Algorithm.

```
from sklearn.linear_model import LinearRegression
```


```
model=LinearRegression()
```

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



▾ LinearRegression ⓘ ?  
 LinearRegression()


```
x_train.shape
```



```
(249, 156)
```

Now we will see coeficient values and the intercept's value

```
model.coef_[0]
```



```
array([ 5.46903952e+03,  3.78426401e+03, -2.39213200e+04,  1.21969522e+04,
        1.43939044e+04,  1.31781684e+04,  1.15530456e+05,  1.08650753e+05,
        1.08376641e+05,  3.78426401e+03,  2.16883204e+04,  2.69669989e+03,
       -7.43147198e+03, -1.37842640e+04, -3.64720797e+03, -3.10609556e+03,
       -4.86294396e+03,  1.83152245e+04, -1.21573599e+03,  6.82537642e+03,
        7.89827363e+04,  5.27670003e+04,  5.35136968e+04,  5.39827363e+04,
        5.89827363e+04,  4.71609048e+04,  5.64964419e+04,  4.83752228e+04,
```

```

5.01984723e+04, 5.77670003e+04, 4.59451688e+04, -1.95291880e+04,
1.18253764e+04, 2.43147198e+03, -1.21573599e+03, -4.86294396e+03,
-1.14898481e+04, 1.33730959e+04, 5.89827363e+04, 8.23603987e+03,
2.56852802e+03, -6.35279203e+03, -3.10786800e+03, -6.82360399e+03,
-6.21573599e+03, -2.65693732e+03, -4.32360399e+03, -2.78643400e+04,
-6.21573599e+03, -1.24314720e+04, -6.35279203e+03, -3.64720797e+03,
1.00000000e+04, -3.10786800e+03, -6.21573599e+03, -5.60786800e+03,
-1.62098132e+03, 3.73857866e+03, -3.35996287e+03, -2.33090100e+03,
-7.43147198e+03, 2.52284267e+03, 7.50000000e+03, 4.50930675e+02,
-6.21573599e+03, -7.20600664e+03, -6.21573599e+03, -6.21573599e+03,
-8.71573599e+03, -7.43147198e+03, -6.00837950e+03, -6.21573599e+03,
-8.78426401e+03, -1.21573599e+03, -3.73857866e+03, 1.81781684e+04,
-5.54822414e+03, 7.65512644e+04, 2.45888319e+04, 4.35136968e+04,
2.75720729e+04, 2.19624325e+04, 3.12175084e+04, 1.93939044e+04,
1.66883204e+04, 3.16883204e+04, 6.82537642e+03, -1.27055841e+04,
7.86507528e+04, 2.59451688e+04, -1.12157360e+04, 5.68832478e+04,
4.06675119e+04, 8.44162392e+03, 2.02030388e+03, 9.72588793e+03,
1.71573599e+04, 2.36472080e+04, 2.82835506e+04, 3.47114201e+04,
3.24350169e+04, 4.23664888e+04, 4.36507528e+04, 4.40559982e+04,
1.19624325e+04, 2.73638302e+04, 2.94120133e+04, 7.90405638e+03,
1.56096404e+04, 3.85125152e+04, 2.29441595e+03, 2.54725844e+04,
3.06096404e+04, 3.71573599e+04, 1.71573599e+04, 4.29743568e+04,
1.56096404e+04, 1.10786800e+04, 4.48664888e+04, 2.97150226e+04,
2.73401013e+04, 1.67550760e+04, 3.48629440e+04, 3.06277493e+04,
3.17586208e+04, 3.36472080e+04, 4.12310901e+04, 2.16883204e+04,
1.72944159e+04, 2.74286273e+04, 2.59451688e+04, 3.10822248e+04,
2.95888319e+04, 9.72588793e+03, 2.75720729e+04, 4.06096404e+04,
3.17568484e+04, 3.45695414e+04, 9.25684840e+03, 4.30428848e+04,
3.27418093e-11, 1.83152245e+04, 3.12157360e+04, 2.30411124e+04,
2.10786800e+04, 3.54725844e+04, 6.88324784e+03, -2.97969612e+03,
-1.37056035e+02, -2.43147198e+03, -2.02741121e+04, 1.57466965e+04,
1.20994885e+04, 8.27670003e+04, 7.27670003e+04, 2.43147198e+03])

```

```
model.intercept_
```

```
→ array([39862.94396483])
```

## ➤ Error and R<sup>2</sup> score value calculation:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

At first we calculate mean absolute error and mean squared error

```
y_pred=model.predict(x_test)
```

```
mean_absolute_error(y_test,y_pred)
```

```
→ 13424.473465628906
```

```
mean_squared_error(y_test,y_pred)
```

```
→ 314529321.79560375
```

To find how well our model is fit we will calculate R<sup>2</sup> score

```
r2_score(y_test,y_pred)
```

```
↵ 0.8574516839044533
```

Our Model's  $R^2$  score value is greater than 0.85 . So we no need to apply any other Algorithm.

## ➤ Model Test by the user

Now we will test our model by user. So in next step we will create user input system.

```
Edu=input("Enter your education label:").lower()
```

```
↵ Enter your education label:phd
```

```
Job=input("Enter your job title:").lower()
```

```
↵ Enter your job title:senior scientist
```

```
Exp=float(input("Enter your experience:"))
```

```
↵ Enter your experience:11
```

```
uid=pd.DataFrame([[Edu,Job,Exp]],columns=["Education Level","Job Title","Years of Experience"])
uid
```

```
↵
```

	Education Level	Job Title	Years of Experience
0	phd	senior scientist	11.0

```
uid_copy=uid.copy()
```

```
uid2_copy=uid.copy()
```

```
uid["Education Level"]=ord.transform(uid[["Education Level"]])
```

```
uid
```

```
↵
```

	Education Level	Job Title	Years of Experience
0	2.0	senior scientist	11.0

```
encode=ohe.transform(uid[["Job Title"]])
```

```
encode_df2=pd.DataFrame(encode,columns=ohe.get_feature_names_out(["Job Title"]))
```

```
encode_df2
```



	Job Title_accountant	Job Title_administrative assistant	Job Title_business analyst	Job Title_business development manager	Job Title_business intelligence analyst	Job Title_
0	0.0	0.0	0.0	0.0	0.0	0.0

1 rows × 155 columns

```
encode_df2.drop("Job Title_accountant",axis=1,inplace=True)
```

```
encode_df2
```



	Job Title_administrative assistant	Job Title_business analyst	Job Title_business development manager	Job Title_business intelligence analyst	Job Title_ceo	Job Title_chief data officer
0	0.0	0.0	0.0	0.0	0.0	0.0

1 rows × 154 columns

```
uid=pd.concat([uid.drop("Job Title",axis=1),encode_df2],axis=1)
```

```
uid
```



	Education Level	Years of Experience	Job Title_administrative assistant	Job Title_business analyst	Job Title_business development manager	Job Title_business intelligence analyst
0	2.0	11.0	0.0	0.0	0.0	0.0

1 rows × 156 columns

```
uid.isnull().sum().sum()
```



```
np.int64(0)
```

```
pred_y=model.predict(uid)
```

```
pred_y
```



```
array([[120000.]])
```

## ➤ Creating Pipeline :

As we seen before in Model Testing by user, the predicted output is very close to actual output. So now we can say that our Model well trained.



But every time we need to preprocess as well as encode our Testing Data. So in next step we will creat a Pipeline using preprocessor and LinearRegression class.

```
from sklearn.base import BaseEstimator,TransformerMixin

class preprocessor (BaseEstimator,TransformerMixin):
    def __init__(self):
        self.ord=OrdinalEncoder(categories=[["bachelor's", "master's", 'phd']])
        self.ohe=OneHotEncoder(handle_unknown="ignore",sparse_output=False)

    def fit(self,x_train,y_train=None):
        self.ord.fit(x_train[["Education Level"]])
        self.ohe.fit(x_train[["Job Title"]])
        return self

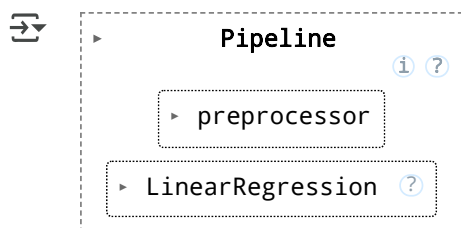
    def transform(self,x):

        x["Education Level"]=self.ord.transform(x[["Education Level"]])
        encode=self.ohe.transform(x[["Job Title"]])
        encode_df=pd.DataFrame(encode,columns=ohe.get_feature_names_out(["Job Title"]))
        encode_df.drop("Job Title_accountant",axis=1,inplace=True)
        x=pd.concat([x.drop("Job Title",axis=1).reset_index(drop=True),encode_df],axis=1)
        return x
```

```
from sklearn.pipeline import Pipeline
```

```
pipe=Pipeline([("preprocessor",preprocessor()),("model",LinearRegression())])
```

```
pipe.fit(x_train_copy,y_train)
```



```
pipe.predict(uid_copy)
```

```
array([[120000.]])
```

### ➤ Save Pipeline as pkl form :

```
import pickle as pkl
```

```
with open("/content/drive/My Drive/Our Project/salary_prediction_model.pkl","wb") as file:
    pkl.dump(pipe,file)
```

## ➤ Result:

After training the model using Linear Regression, it was evaluated on the test dataset using the  $R^2$  score, a metric that measures how well the model predicts the target variable.

The model achieved an  $R^2$  score greater than 0.85, indicating a strong linear relationship between the input features (Education Level, Job Title, and Years of Experience) and the predicted Salary. This means the model was able to explain over 85% of the variance in salary values based on the given features.

Such a performance demonstrates the model's effectiveness and reliability in salary prediction tasks.

## ➤ How to Use the Model :

### ❖ Method 1 -> Using the Pretrained .pkl Model :

1. First, clone the repository by running the following command:

```
git clone https://github.com/trio-rds-tensors/Salary\_Prediction\_Model.git
```

2. Then, install the required Python libraries: NumPy, Pandas, and Scikit-learn using the command:

```
pip install numpy pandas scikit-learn
```

3. After setting up the environment, you can load and use the model as follows:

Import the preprocessing module and the pickle library.

Load the trained model using `pickle.load()`.

Use the `predict()` method of the loaded model by passing the required input features (Education Level, Job Title, and Years of Experience).

For example: `predicted_salary = model.predict([[Edu, Job, Exp]])`

### ❖ Method 2 -> Creating and Using the Model Manually:

1. Clone the same repository:

```
git clone https://github.com/trio-rds-tensors/Salary\_Prediction\_Model.git
```

2. Import the custom model class from the main Python file.

Create an instance of the model and call the `predict()` method with appropriate inputs.

For example:

```
from Salary_Prediction_Model import Salary
model = Salary()
model.predict([Edu, Job, Exp])
```

**For more details visit :** [https://github.com/trio-rds-tensors/Salary\\_Prediction\\_Model.git](https://github.com/trio-rds-tensors/Salary_Prediction_Model.git)

## ➤ Conclusion:

The Salary Prediction Model developed in this project successfully demonstrates how machine learning, specifically Linear Regression, can be applied to predict salaries based on key features such as education level, job title, and years of experience. By preprocessing the data effectively and focusing on the most relevant variables, the model achieved a strong  $R^2$  score exceeding 0.85, indicating high predictive accuracy.

This model can serve as a helpful tool for job seekers, HR professionals, and employers to make data-driven decisions regarding fair compensation. The use of Python libraries such as Scikit-learn, Pandas, and Pickle enabled smooth development, training, evaluation, and saving of the model for future use.

Future improvements could involve using more advanced models like ensemble methods, expanding the dataset, or building a user-friendly interface to make the tool more accessible.