Komal Kumari

Internsip Offer Letter from Mentorness

Project Name: Salary Predictions of Data Professions

```
Welcome to the Machine Learning Internship, focused on predicting the salaries of data professionals. In
this project, you will dive into the world of regression tasks and gain hands-on experience in data
analysis, feature engineering, and machine learning model development. The goal is to predict the
salaries of data professionals based on a rich dataset.
```

Problem Statement:

```
Salaries in the field of data professions vary widely based on factors such as experience, job role, and
performance. Accurately predicting salaries for data professionals is essential for both job seekers and
employers.
```

Your Mission:

1. Exploratory Data Analysis (EDA):
2. Feature Engineering:
3. Data Preprocessing:
4. Machine Learning Model Development:
5. Model Evaluation:
6. ML Pipelines and Model Deployment:
7. Recommendations:

```
# step-1 Load the liabraies
```

```
# step-1 Load the liabraies
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

step 2 Data loading and inspection phase

In [18]:
```
df = pd.read_csv('Salary Prediction of Data Professions.csv')
df
```

Out[18]:

	FIRST NAME	LAST NAME	SEX	DOJ	CURRENT DATE	DESIGNATION	AGE	SALARY	UNIT	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
0	TOMASA	ARMEN	F	5-18-2014	01-07-2016	Analyst	21.0	44570	Finance	24.0	6.0	2.0	0
1	ANNIE	NaN	F	NaN	01-07-2016	Associate	NaN	89207	Web	NaN	13.0	NaN	7
2	OLIVE	ANCY	F	7-28-2014	01-07-2016	Analyst	21.0	40955	Finance	23.0	7.0	3.0	0
3	CHERRY	AQUILAR	F	04-03-2013	01-07-2016	Analyst	22.0	45550	IT	22.0	8.0	3.0	0
4	LEON	ABOULAHOUD	M	11-20-2014	01-07-2016	Analyst	NaN	43161	Operations	27.0	3.0	NaN	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2634	KATHERINE	ALSDON	F	6-28-2011	01-07-2016	Senior Manager	36.0	185977	Management	15.0	15.0	5.0	10
2635	LOUISE	ALTARAS	F	1-14-2014	01-07-2016	Analyst	23.0	45758	IT	17.0	13.0	2.0	0
2636	RENEE	ALVINO	F	1-23-2014	01-07-2016	Analyst	21.0	47315	Web	29.0	1.0	5.0	0
2637	TERI	ANASTASIO	F	3-17-2014	01-07-2016	Analyst	24.0	45172	Web	23.0	7.0	3.0	1
2638	GREGORY	ABARCA	M	9-18-2014	01-07-2016	Analyst	24.0	49176	Marketing	17.0	13.0	2.0	2

In [7]:
```
# data inspection
```

In [19]:
```
df.shape
```

Out[19]:
```
(2639, 13)
```

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

Out[20]:

	FIRST NAME	LAST NAME	SEX	DOJ	CURRENT DATE	DESIGNATION	AGE	SALARY	UNIT	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
0	TOMASA	ARMEN	F	5-18-2014	01-07-2016	Analyst	21.0	44570	Finance	24.0	6.0	2.0	0
1	ANNIE	NaN	F	NaN	01-07-2016	Associate	NaN	89207	Web	NaN	13.0	NaN	7
2	OLIVE	ANCY	F	7-28-2014	01-07-2016	Analyst	21.0	40955	Finance	23.0	7.0	3.0	0
3	CHERRY	AQUILAR	F	04-03-2013	01-07-2016	Analyst	22.0	45550	IT	22.0	8.0	3.0	0
4	LEON	ABOULAHOUD	M	11-20-2014	01-07-2016	Analyst	NaN	43161	Operations	27.0	3.0	NaN	3

Insigh ***we have a lot of categorical values, this mean we have to do encoding

In [21]:
```
# data information
```

In [23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2639 entries, 0 to 2638
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   FIRST NAME      2639 non-null   object
 1   LAST NAME       2637 non-null   object
 2   SEX             2639 non-null   object
 3   DOJ             2638 non-null   object
 4   CURRENT DATE    2639 non-null   object
 5   DESIGNATION     2639 non-null   object
 6   AGE             2636 non-null   float64
 7   SALARY          2639 non-null   int64
                    2639 non-null   object
                    2636 non-null   float64
```

ed2c061f524f43c8e0ca54bd2...

```
 12  PAST EXP           2639 non-null   int64
dtypes: float64(4), int64(2), object(7)
memory usage: 268.2+ KB
```

In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
161
```

In [24]:
```
#dropna
df = df.dropna()
```

In [25]:
```
df.isnull().sum()
```

Out[25]:
```
FIRST NAME          0
LAST NAME           0
SEX                 0
DOJ                 0
CURRENT DATE        0
DESIGNATION         0
AGE                 0
SALARY              0
UNIT                0
LEAVES USED         0
LEAVES REMAINING    0
RATINGS             0
PAST EXP            0
dtype: int64
```

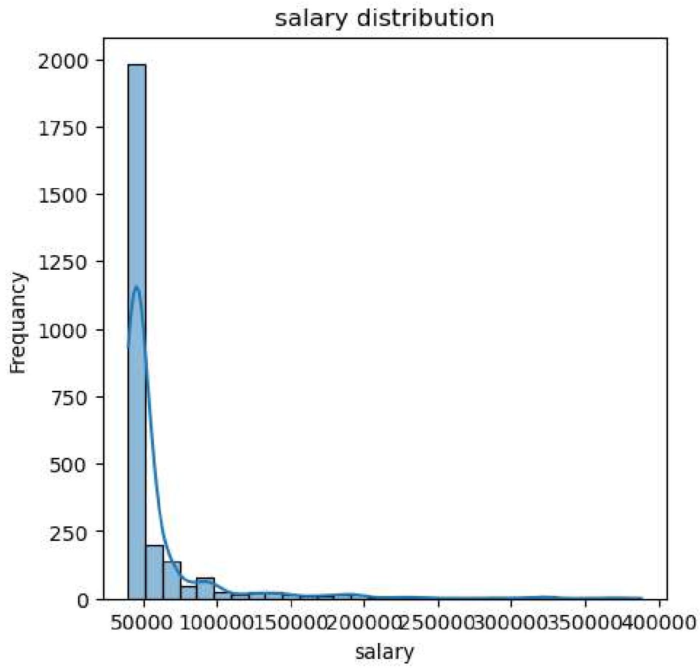
In [26]:
```
df.describe()
```

Out[26]:

	AGE	SALARY	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
count	2631.000000	2631.000000	2631.000000	2631.000000	2631.000000	2631.000000
mean	24.754846	58117.644242	22.498670	7.501330	3.486507	1.562904
std	3.904705	36867.732515	4.603014	4.603014	1.114248	2.725973
min	21.000000	40001.000000	15.000000	0.000000	2.000000	0.000000
25%	22.000000	43418.000000	19.000000	4.000000	2.000000	0.000000
50%	24.000000	46783.000000	22.000000	8.000000	3.000000	1.000000
75%	25.000000	51401.500000	26.000000	11.000000	4.000000	2.000000
max	45.000000	388112.000000	30.000000	15.000000	5.000000	23.000000

In [27]:
```
# Histogram for Salary Distribution
plt.figure(figsize=(5,5))
sns.histplot(df['SALARY'], bins=30, kde=True)
plt.title('salary distribution')
plt.xlabel('salary')
plt.ylabel('Frequancy')
plt.show()

# Boxplot for Salary by Designation
plt.figure(figsize=(6, 6))
sns.boxplot(x='DESIGNATION', y='SALARY', data=df)
plt.title('Salary by Designation')
plt.xticks(rotation=45)
plt.show()
```



DESIGNATION

In [13]:
```
from scipy import stats

# Calculate z-scores for Salary
z_scores = stats.zscore(df['SALARY'])

# Identify outliers (threshold z-score > 3 or < -3)
outliers = df['SALARY'][np.abs(z_scores) > 3]

print("Outliers based on Z-score:")
print(outliers)
```

```
Outliers based on Z-score:
41      175497
73      193621
114     189435
160     323196
166     388112
         ...
2546    213987
2575    195985
2584    187837
2595    194701
2634    185977
Name: SALARY, Length: 76, dtype: int64
```

In [14]:
```
# Calculate IQR for Salary
Q1 = df['SALARY'].quantile(0.25)
Q3 = df['SALARY'].quantile(0.75)
IQR = Q3 - Q1

# Identify outliers based on IQR
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_iqr = df[(df['SALARY'] < lower_bound) | (df['SALARY'] > upper_bound)]['SALARY']

print("Outliers based on IQR:")
print(outliers_iqr)
```

```
Outliers based on IQR:
8        63478
32       68295
```

```
Outliers based on IQR:
8        63478
32       68295
33       73397
41      175497
45       96378
         ...
2603    154120
2608     86705
2611     94345
2618     66661
2634    185977
Name: SALARY, Length: 452, dtype: int64
```

In [15]:
```python
df.shape
```

Out[15]:
```
(2631, 13)
```

2. Feature Engineering

Creating New Features

In [28]:
```python
from datetime import datetime

# Convert dates to datetime
df['DOJ'] = pd.to_datetime(df['DOJ'])
df['CURRENT DATE'] = pd.to_datetime(df['CURRENT DATE'])

# Calculate tenure
df['TENURE'] = (df['CURRENT DATE'] - df['DOJ']).dt.days / 365

# Display the new feature
print(df[['DOJ', 'CURRENT DATE', 'TENURE']].head())
```

```
         DOJ CURRENT DATE    TENURE
0 2014-05-18   2016-01-07  1.641096
2 2014-07-28   2016-01-07  1.446575
3 2013-04-03   2016-01-07  2.764384
6 2013-09-02   2016-01-07  2.347945
8 2014-06-29   2016-01-07  1.526027
```

In [29]:
```python
# Log transformation for skewed features (if needed)
df['LOG_SALARY'] = df['SALARY'].apply(lambda x: np.log(x) if x > 0 else 0)

# Display the transformed feature
print(df[['SALARY', 'LOG_SALARY']].head())
```

```
   SALARY  LOG_SALARY
0   44570   10.704816
2   40955   10.620229
3   45550   10.726566
6   40339   10.605074
8   63478   11.058449
```

3. Data Preprocessing

In [30]:
```python
# Impute missing values (example: with median)
df['PAST EXP'].fillna(df['PAST EXP'].median(), inplace=True)

# Drop rows with missing target values
df.dropna(subset=['SALARY'], inplace=True)

# Check the updated dataset for missing values
print(df.isnull().sum())
```

```
FIRST NAME          0
LAST NAME           0
SEX                 0
DOJ                 0
CURRENT DATE        0
DESIGNATION         0
AGE                 0
SALARY              0
UNIT                0
LEAVES USED         0
LEAVES REMAINING    0
RATINGS             0
PAST EXP            0
TENURE              0
LOG_SALARY          0
dtype: int64
```

In [35]:
```python
# One-Hot Encoding for categorical features
df_encoded = pd.get_dummies(df, columns=['SEX', 'DESIGNATION', 'UNIT'], drop_first=True)
```

In [36]:
```python
# Display the encoded dataset
print(df_encoded.head())
```

```
  FIRST NAME LAST NAME        DOJ CURRENT DATE   AGE  SALARY  LEAVES USED  \
0     TOMASA     ARMEN 2014-05-18   2016-01-07  21.0   44570         24.0
2      OLIVE      ANCY 2014-07-28   2016-01-07  21.0   40955         23.0
3     CHERRY   AQUILAR 2013-04-03   2016-01-07  22.0   45550         22.0
6     ELLIOT    AGULAR 2013-09-02   2016-01-07  22.0   40339         19.0
8      KATHY     ALSOP 2014-06-29   2016-01-07  28.0   63478         20.0

   LEAVES REMAINING  RATINGS  PAST EXP  ...  DESIGNATION_Associate  \
0               6.0      2.0         0  ...                  False
2               7.0      3.0         0  ...                  False
3               8.0      3.0         0  ...                  False
6              11.0      5.0         0  ...                  False
8              10.0      3.0         1  ...                  False

   DESIGNATION_Director  DESIGNATION_Manager  DESIGNATION_Senior Analyst  \
0                 False                False                       False
2                 False                False                       False
3                 False                False                       False
6                 False                False                       False
8                 False                False                        True

   DESIGNATION_Senior Manager  UNIT_IT  UNIT_Management  UNIT_Marketing  \
0                       False    False            False           False
2                       False    False            False           False
3                       False     True            False           False
6                       False    False            False            True
8                       False    False            False           False

   UNIT_Operations  UNIT_Web
0            False     False
2            False     False
3            False     False
6            False     False
8             True     False

[5 rows x 23 columns]
```

Feature Scaling

In [37]:
```
from sklearn.preprocessing import StandardScaler

# Select numerical features for scaling
num_features = ['AGE', 'LEAVES USED', 'LEAVES REMAINING', 'RATINGS', 'PAST EXP', 'TENURE']
```

In [38]:
```
# Initialize the scaler
scaler = StandardScaler()

# Apply scaling
df_encoded[num_features] = scaler.fit_transform(df_encoded[num_features])
```

4. Machine Learning Model Development

```
#Splitting the Data
```

In [39]:
```
from sklearn.model_selection import train_test_split

# Define features (X) and target (y)
X = df_encoded.drop(columns=['FIRST NAME', 'LAST NAME', 'DOJ', 'CURRENT DATE', 'SALARY', 'LOG_SALARY'])
y = df_encoded['SALARY']
```

In [40]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f'Training set size: {X_train.shape[0]}')
print(f'Testing set size: {X_test.shape[0]}')
```

```
Training set size: 2104
Testing set size: 527
```

Training Models

In [41]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Initialize models
lr_model = LinearRegression()
rf_model = RandomForestRegressor(random_state=42)

# Train Linear Regression
lr_model.fit(X_train, y_train)

# Train Random Forest Regressor
rf_model.fit(X_train, y_train)
```

Out[41]:
```
RandomForestRegressor(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

5. Model Evaluation

Evaluating Models

In [43]:
```
# Make predictions
y_pred_lr = lr_model.predict(X_test)
y_pred_rf = rf_model.predict(X_test)
```

In [45]:
```
# Define evaluation metrics
def evaluate_model(y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    return mae, mse, rmse, r2

# Evaluate Linear Regression
lr_metrics = evaluate_model(y_test, y_pred_lr)
print(f'Linear Regression - MAE: {lr_metrics[0]}, MSE: {lr_metrics[1]}, RMSE: {lr_metrics[2]}, R2: {lr_metrics[3]}')
```

```python
# Evaluate Linear Regression
lr_metrics = evaluate_model(y_test, y_pred_lr)
print(f'Linear Regression - MAE: {lr_metrics[0]}, MSE: {lr_metrics[1]}, RMSE: {lr_metrics[2]}, R2: {lr_metrics[3]}')

# Evaluate Random Forest
rf_metrics = evaluate_model(y_test, y_pred_rf)
print(f'Random Forest - MAE: {rf_metrics[0]}, MSE: {rf_metrics[1]}, RMSE: {rf_metrics[2]}, R2: {rf_metrics[3]}')
```

```
Linear Regression - MAE: 4026.7808774035466, MSE: 46753532.101031, RMSE: 6837.655453518479, R2: 0.9626399702009055
Random Forest - MAE: 4123.2525553447185, MSE: 54393199.11467526, RMSE: 7375.174514184411, R2: 0.9565352295650935
```

6. ML Pipelines and Model Deployment

In [48]:
```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor
import joblib
```

In [49]:
```python
# Sample DataFrame setup
data = {
    'AGE': [25, 32, 47],
    'PAST EXP': [2, 10, 20],
    'SEX': ['M', 'F', 'M'],
    'DESIGNATION': ['Data Scientist', 'Data Engineer', 'Analyst'],
    'UNIT': ['R&D', 'IT', 'Finance']
}
X_train = pd.DataFrame(data)
```

In [50]:
```python
X_train
```

Out[50]:

	AGE	PAST EXP	SEX	DESIGNATION	UNIT
0	25	2	M	Data Scientist	R&D
1	32	10	F	Data Engineer	IT
2	47	20	M	Analyst	Finance

In [51]:
```python
# Example target values (make sure this has the same length as X_train rows)
y_train = [50000, 120000, 70000]
```

In [52]:
```python
# Print the shapes to verify
print("Shape of X_train:", X_train.shape)
print("Length of y_train:", len(y_train))
```

```
Shape of X_train: (3, 5)
Length of y_train: 3
```

In [53]:
```python
# Define your columns
numerical_features = ['AGE', 'PAST EXP']
categorical_features = ['SEX', 'DESIGNATION', 'UNIT']

# Check if the columns are present in X_train
print("Numerical Features:", set(numerical_features).difference(X_train.columns))
print("Categorical Features:", set(categorical_features).difference(X_train.columns))
```

```
Numerical Features: set()
Categorical Features: set()
```

In [54]:
```python
# Ensure X_train is a DataFrame with correct columns
print(type(X_train))
print(X_train.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
Index(['AGE', 'PAST EXP', 'SEX', 'DESIGNATION', 'UNIT'], dtype='object')
```

In [56]:
```python
# Define the transformations for numerical features
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
```

```
<class 'pandas.core.frame.DataFrame'>
Index(['AGE', 'PAST EXP', 'SEX', 'DESIGNATION', 'UNIT'], dtype='object')
```

In [56]:
```
# Define the transformations for numerical features
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
```

In [57]:
```
# Define the transformations for categorical features
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

In [58]:
```
# Combine the transformers into a preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

In [59]:
```
# Define the model
model = RandomForestRegressor(n_estimators=100, random_state=0)
```

In [60]:
```
# Create the pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', model)
])

# Fit the pipeline
pipeline.fit(X_train, [50000, 120000, 70000])  # Example target values
```

In [60]:
```
# Create the pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', model)
])

# Fit the pipeline
pipeline.fit(X_train, [50000, 120000, 70000])  # Example target values
```

Out[60]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer()),
                                                                  ('scaler',
                                                                   StandardScaler())]),
                                                  ['AGE', 'PAST EXP']),
                                                 ('cat',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEncoder(handle_unknown='ignore'))]),
                                                  ['SEX', 'DESIGNATION',
                                                   'UNIT'])])),
                ('model', RandomForestRegressor(random_state=0))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [61]:
```
# Make predictions
print(pipeline.predict(X_train))
```

```
[ 67700. 102900.  76000.]
```

In [62]:
```
# Create a pipeline that combines the preprocessor with the model
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', RandomForestRegressor(n_estimators=100, random_state=42))
])
```

In [63]:
```
# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)
```

In [62]:
```
# Create a pipeline that combines the preprocessor with the model
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', RandomForestRegressor(n_estimators=100, random_state=42))
])
```

In [63]:
```
# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)
```

Out[63]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer()),
                                                                  ('scaler',
                                                                   StandardScaler())]),
                                                  ['AGE', 'PAST EXP']),
                                                 ('cat',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEncoder(handle_unknown='ignore'))]),
                                                  ['SEX', 'DESIGNATION',
                                                   'UNIT'])])),
                ('model', RandomForestRegressor(random_state=42))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [64]:
```

# Save the pipeline for future use
import joblib
joblib.dump(pipeline, 'salary_prediction_pipeline.pkl')
```

Out[64]:
```
['salary_prediction_pipeline.pkl']
```

In [65]:
```
loaded_pipeline = joblib.load('salary_prediction_pipeline.pkl')
```

In [66]:
```
# Load new data into a DataFrame (example new data)
new_data = {
    'AGE': [30, 40],
```

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [64]:
```python
# Save the pipeline for future use
import joblib
joblib.dump(pipeline, 'salary_prediction_pipeline.pkl')
```

Out[64]:
```
['salary_prediction_pipeline.pkl']
```

In [65]:
```python
loaded_pipeline = joblib.load('salary_prediction_pipeline.pkl')
```

In [66]:
```python
# Load new data into a DataFrame (example new data)
new_data = {
    'AGE': [30, 40],
    'PAST EXP': [5, 15],
    'SEX': ['F', 'M'],
    'DESIGNATION': ['Analyst', 'Manager'],
    'UNIT': ['Marketing', 'Finance']
}
X_new = pd.DataFrame(new_data)

# Predict salaries using the loaded pipeline
new_predictions = loaded_pipeline.predict(X_new)
print("Predictions on new data:", new_predictions)
```

```
Predictions on new data: [87500. 76600.]
```

In [ ]: