SALARY PREDICTION - Mentorness

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.

Exploratory Data Analysis (EDA):

	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

2639 rows × 13 columns

```
Duplicates: 161

FIRST NAME            0
LAST NAME             2
SEX                   0
DOJ                   1
CURRENT DATE          0
DESIGNATION           0
AGE                   3
SALARY                0
UNIT                  0
LEAVES USED           3
LEAVES REMAINING      2
RATINGS               2
PAST EXP              0
dtype: int64
```

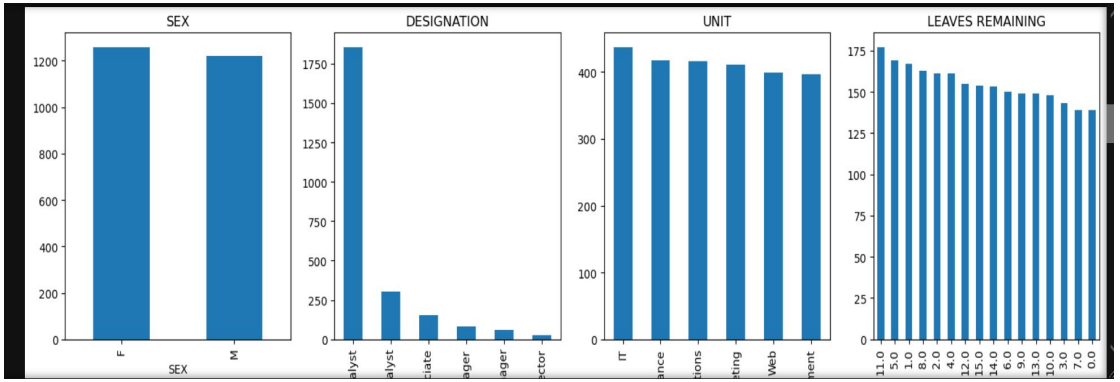
The data set comprised **2639 rows** and **13 columns**.
161 duplicates are dropped.
Missing values were filled with **mean** and **median** for **numeric** and **categorical** features respectively.

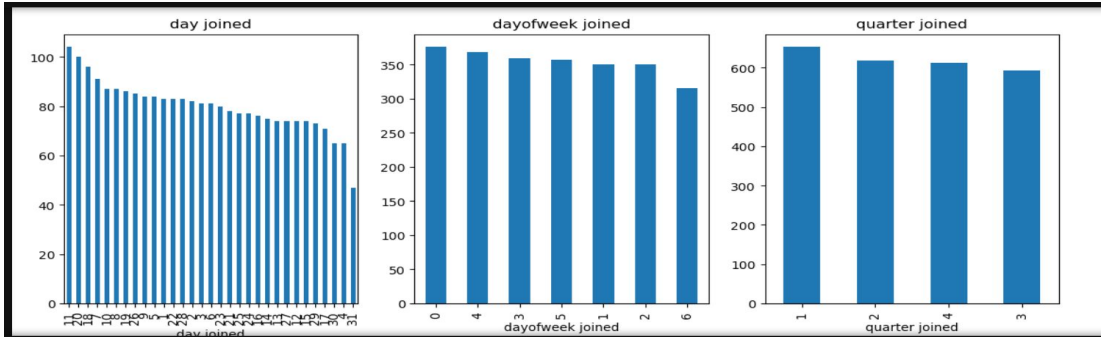
Features Distribution:

The diagram shows the distribution of features in the dataset.

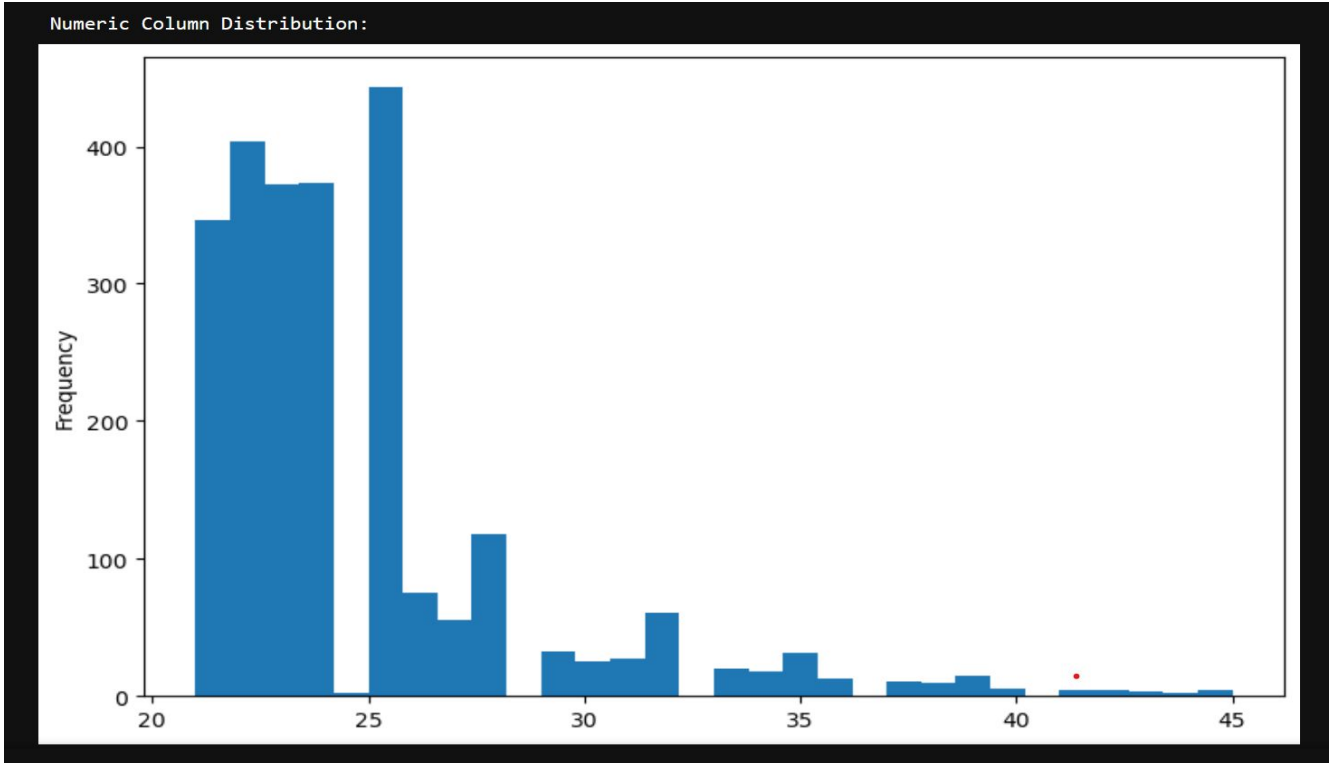
New features such as **“day joined”, “dayofweek joined”, “quarter joined”, and“month joined”** were also created from the **“Date Joined”** column.

Other columns that have just one value were also drop(**“year joined”**,)





Features Distribution:

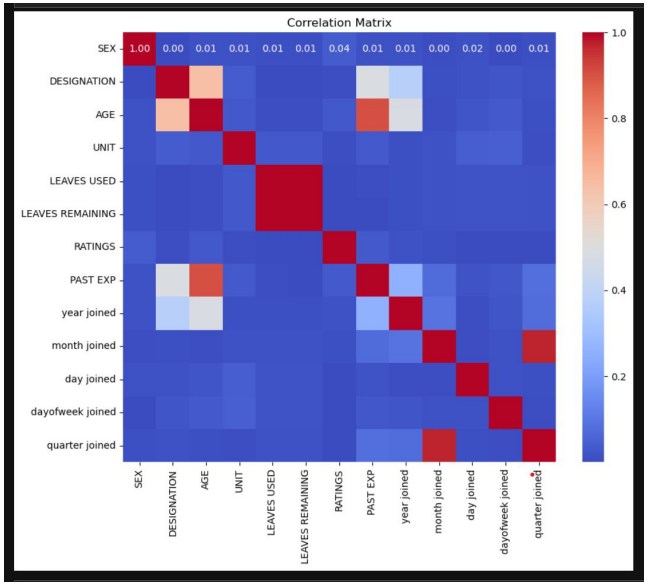


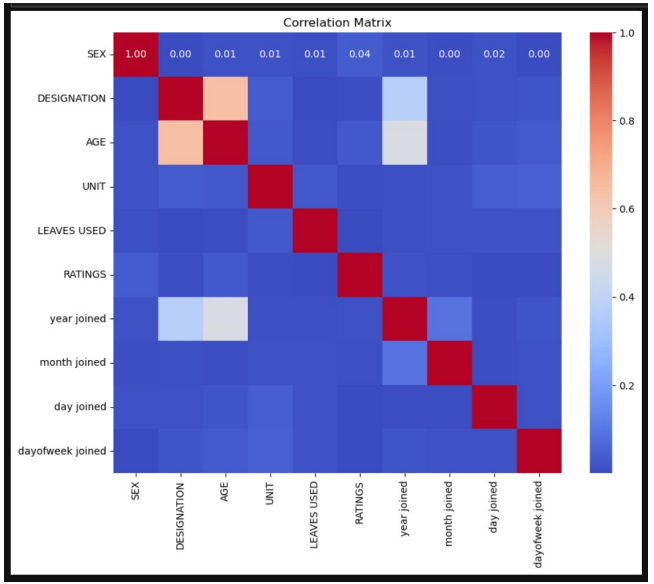
The relationship between the features and the label.

Categorical features were encoded with **LabelEncoder()**, and respective mapped fugures were noted down for **deployment.**

```
DESIGNATION Mapping:
{'Analyst': 0, 'Associate': 1, 'Director': 2, 'Manager': 3, 'Senior Analyst': 4, 'Senior Manager': 5}

UNIT Mapping:
{'Finance': 0, 'IT': 1, 'Management': 2, 'Marketing': 3, 'Operations': 4, 'Web': 5}

LEAVES REMAINING Mapping:
{0.0: 0, 1.0: 1, 2.0: 2, 3.0: 3, 4.0: 4, 5.0: 5, 6.0: 6, 7.0: 7, 8.0: 8, 9.0: 9, 10.0: 10, 11.0: 11, 12.0: 12, 13.0: 13, 14.0: 14, 15.0: 15}

LEAVES USED Mapping:
{15.0: 0, 16.0: 1, 17.0: 2, 18.0: 3, 19.0: 4, 20.0: 5, 21.0: 6, 22.0: 7, 23.0: 8, 24.0: 9, 25.0: 10, 26.0: 11, 27.0: 12, 28.0: 13, 29.0: 14, 30.0: 15}

RATINGS Mapping:
{2.0: 0, 3.0: 1, 4.0: 2, 5.0: 3}

year joined Mapping:
{2009: 0, 2010: 1, 2011: 2, 2012: 3, 2013: 4, 2014: 5, 2015: 6}

month joined Mapping:
{1: 0, 2: 1, 3: 2, 4: 3, 5: 4, 6: 5, 7: 6, 8: 7, 9: 8, 10: 9, 11: 10, 12: 11}

day joined Mapping:
{1: 0, 2: 1, 3: 2, 4: 3, 5: 4, 6: 5, 7: 6, 8: 7, 9: 8, 10: 9, 11: 10, 12: 11, 13: 12, 14: 13, 15: 14, 16: 15, 17: 16, 18: 17, 19: 18, 20: 19, 21: 20, 22
: 21, 23: 22, 24: 23, 25: 24, 26: 25, 27: 26, 28: 27, 29: 28, 30: 29, 31: 30}

dayofweek joined Mapping:
{0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6}

quarter joined Mapping:
{1: 0, 2: 1, 3: 2, 4: 3}
```

In training the model, one feature that correlated up to **0.9%** with another was dropped. Hence reducing the features to **10.**





Different regressors were trained, and each model's metrics such as it **error** and **fitting** were tested.

For a **threshold** of **100** for the **Mean Absolute Error** of the **train set** to the **test set**, each model was **overfitting.**

Regressor	Train MAE	Train MSE	Train RMSE	Train R2	Test MAE	Test MSE	Test RMSE	Test R2	Train Time
Decision_tree	1.519435	2.286749e+03	47.819966	0.999998	5152.022177	9.926514e+07	9963.189126	0.893725	0.020230
Random_forest	1681.071982	1.115315e+07	3339.633718	0.992333	3812.598068	4.130213e+07	6426.673084	0.955781	1.563744
Ada_boost	99.184755	9.522410e+04	308.584024	0.999935	4050.532258	5.351228e+07	7315.208782	0.942709	0.878919
Bagging	1804.437052	1.314784e+07	3625.995051	0.990962	3939.708266	4.956635e+07	7040.337010	0.946934	0.158583
Gradient_boost	3603.901417	2.534993e+07	5034.871799	0.982574	3969.907198	7.592531e+07	8713.513322	0.918713	0.521242

```
#Checking for overfitting
threshold = 100
checking_result_fitting(all_reg_model, threshold=threshold)
```

```
Decision_tree: is overfitting.

Random_forest: is overfitting.

Ada_boost: is overfitting.

Bagging: is overfitting.

Gradient_boost: is overfitting.
```

The first image shows a **Cross Validation Score** of each **Regressor** with **GradientBoostingRegressor** performing much better than other regressors.

A **Randomized Search Cross Validation** was also made to check the **best parameters** of each model. The **hyperparameter** given were delibrately to treat the issue of **overfitting** for each model.

Regressor	CV R-Squared	CV MAE	CV MSE
Decision_tree	0.893389	-5708.452097	-1.470969e+08
Random_forest	0.943948	-4454.371749	-7.571669e+07
Ada_boost	0.935440	-4786.439548	-9.411551e+07
Bagging	0.940500	-4605.520057	-8.523196e+07
Gradient_boost	0.943891	-4434.983520	-7.641566e+07

```
# Evaluating Random Forest Regressor's new parameters
np.random.seed(42)

rf_improved     = RandomForestRegressor(max_depth=7, max_features=None, max_leaf_nodes=9, min_impurity_decrease=0.4,
                                        min_samples_leaf=2, min_samples_split=2, n_estimators=57)
ada_improved    = AdaBoostRegressor(DecisionTreeRegressor(max_depth=5, min_samples_leaf=13,min_samples_split=11),
                                    learning_rate=0.029724472632554504, n_estimators=72)
bagging_improved = BaggingRegressor(DecisionTreeRegressor(min_samples_split=5, min_samples_leaf=4, max_depth=15),
                                    n_estimators=100,  max_samples=0.8, max_features=0.8, bootstrap=True)
d_tree_improved = DecisionTreeRegressor(criterion='poisson', max_depth=4, max_features=None, max_leaf_nodes=22, min_impurity_decrease=0.5,
                                        min_samples_leaf=4, min_samples_split=11, splitter='best')


best_params_models = {"Random_forest" : rf_improved,
                      "Ada Boosst" : ada_improved,
                      "Bagging Regressor" : bagging_improved,
                      "Decision Tree" : base_model_tree,
                      "Gradient_boost" : gbr}
```

With each model's **best parameters**, the **Gradient Boosting Regressor** came to be the model with the **lowest overfitting** and a resonable **MAE** and **training time.**

Regressor	Train MAE	Train MSE	Train RMSE	Train R2	Test MAE	Test MSE	Test RMSE	Test R2	Train Time
Random_forest	4154.699709	4.904271e+07	7003.049804	0.966287	3716.653714	4.063589e+07	6374.628630	0.956495	0.282188
Ada Boosst	3831.144556	3.413631e+07	5842.628971	0.976534	3848.007156	4.653898e+07	6821.948277	0.950175	0.492162
Bagging Regressor	3337.732930	4.428626e+07	6654.792363	0.969557	3750.177980	3.884486e+07	6232.564727	0.958412	0.867655
Decision Tree	1.519435	2.286749e+03	47.819966	0.999998	5112.193548	9.927530e+07	9963.699011	0.893714	0.025541
Gradient_boost	3603.901417	2.534993e+07	5034.871799	0.982574	3972.698573	7.545495e+07	8686.480619	0.919217	0.397350

```
# Testing for overfitting models

threshold = 400
checking_result_fitting(tuned_model_metrics, threshold=threshold)
```

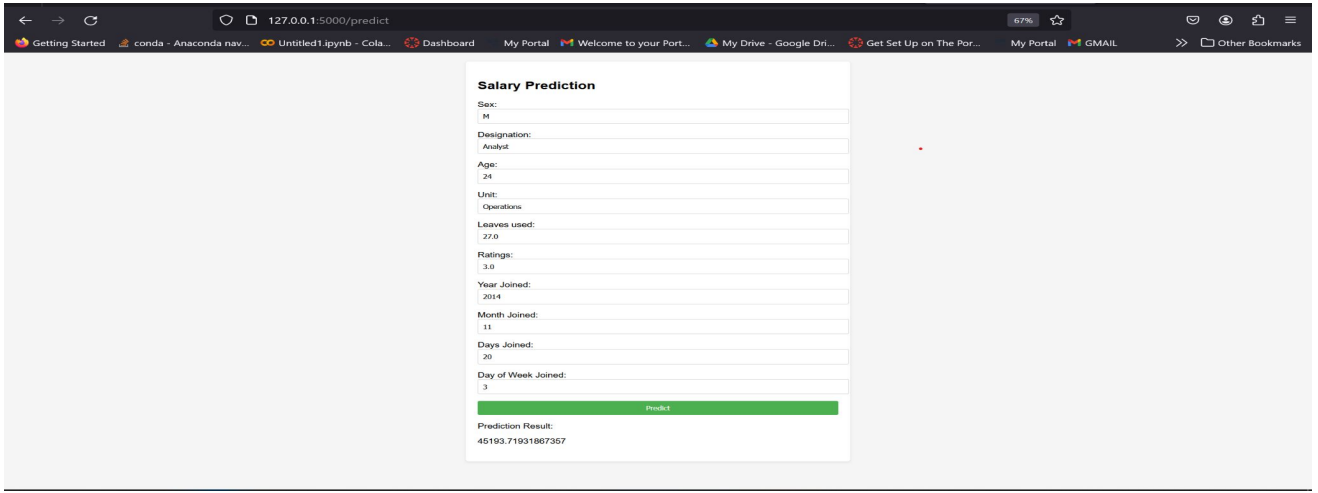
```
Random_forest: is underfitting.
Ada Boosst: is performing well.

Bagging Regressor: is overfitting.

Decision Tree: is overfitting.

Gradient_boost: is performing well.
```

Gradient Boosting Regressor with the **best parameters** was my final prediction.

I created a **end-to-end pipeline** for my work and **deployed** the model using **Flask and HTML.**

```
np.random.seed(42)
model = GradientBoostingRegressor(learning_rate=0.05, max_depth=4, max_features=None, min_samples_leaf=1,
                                  min_samples_split=10, n_estimators=100, random_state=42)
model.fit(X_train, y_train)
print(f"Accuracy: {round(model.score(X_test, y_test) * 100, 2)}%")
```

```
Accuracy: 95.14%
```



Recommendation

Data Collection - The model need to be trained with more dataset and more features to get a good confidence rate for prediction..