Capstone Project-2
(Employee Churn Prediction)
Group-11

F-1295 Victor
F-1380 Habip
F-1470 Aziz
F-1491 Brian
F-1328 Fırat
F-1382 Halil

Table of Contents

Introduction to Employee Churn

- In customer churn, you can predict who and when a customer will stop buying.
- Employee churn is similar to customer churn.
- It was found that employee churn will be affected by age, tenure, pay, job satisfaction, salary, working conditions, growth potential and employee’s perceptions of fairness.
- Some other variables such as gender, ethnicity, education, and marital status were essential factors in the prediction of employee churn.
- In some cases such as the employee with niche skills are harder to replace.
- Acquiring new employees as a replacement has its costs such as hiring costs and training costs.
- Organizations tackle this problem by applying machine learning techniques to predict employee churn, which helps them in taking necessary actions.

Introduction to Employee Churn

- Business chooses the employee to hire someone while in marketing you don't get to choose your customers.
- Employees will be the face of your company, and collectively, the employees produce everything your company does.
- Losing a customer affects revenues and brand image. Acquiring new customers is difficult and costly compared to retain the existing customer. Employee churn also painful for companies an organization. It requires time and effort in finding and training a replacement.
- Employee churn has unique dynamics compared to customer churn. It helps us in designing better employee retention plans and improving employee satisfaction. Data science algorithms can predict the future churn.

Key Takeaways and Assumptions

- A Clean DF (How lucky we are!🙏).
- There are no duplicated value.
- There are remarkable amount of niche employees.
- Clustering does not work for our project.
- XGBoost Model
- Model Deployment —> Streamlit

EDA

1.

skimpy summary

Data Summary

dataframe	Values
Number of rows	14999
Number of columns	10

Data Types

Column Type	Count
int32	6
float64	2
string	2

number

column_name	NA	NA %	mean	sd	p0	p25	p75	p100	hist
satisfaction	0	0	0.61	0.25	0.09	0.44	0.82	1	
last_eval	0	0	0.72	0.17	0.36	0.56	0.87	1	
num_of_projects	0	0	3.8	1.2	2	3	5	7	
avg_monthly_hours	0	0	200	50	96	160	240	310	
experience	0	0	3.5	1.5	2	3	4	10	
work_accident	0	0	0.14	0.35	0	0	0	1	
promotion	0	0	0.021	0.14	0	0	0	1	
left	0	0	0.24	0.43	0	0	0	1	

string

column_name	NA	NA %	words per row	total words
department	0	0	1	15000
salary	0	0	1	15000

End

2. **We chose not to delete duplicated rows**

3. **"groupby" function by ["work_accident", "promotion", "department", "salary", "left"]**

```
df.groupby("work_accident").mean() # promotion, left
```

work_accident	satisfaction	last_eval	num_of_projects	avg_monthly_hours	experience	promotion	left
0	0.607	0.717	3.805	201.259	3.497	0.019	0.265
1	0.648	0.713	3.789	199.818	3.506	0.035	0.078

Then, let's analyze employees according to their promotion status.

```
df.groupby("promotion").mean()  # work accident, left
```

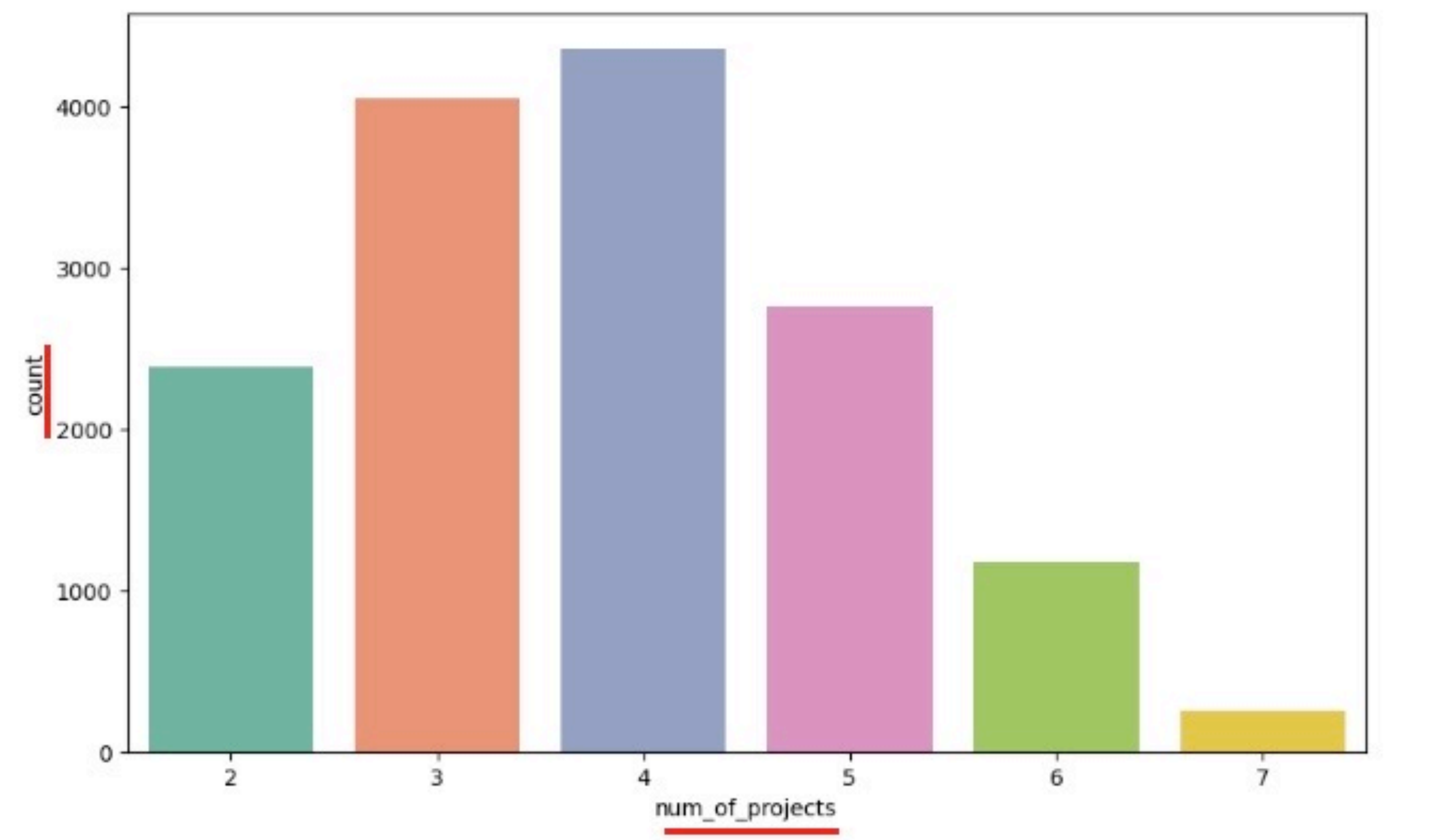
promotion	satisfaction	last_eval	num_of_projects	avg_monthly_hours	experience	work_accident	left
0	0.612	0.716	3.804	201.076	3.484	0.143	0.242
1	0.656	0.706	3.752	199.850	4.166	0.238	0.060

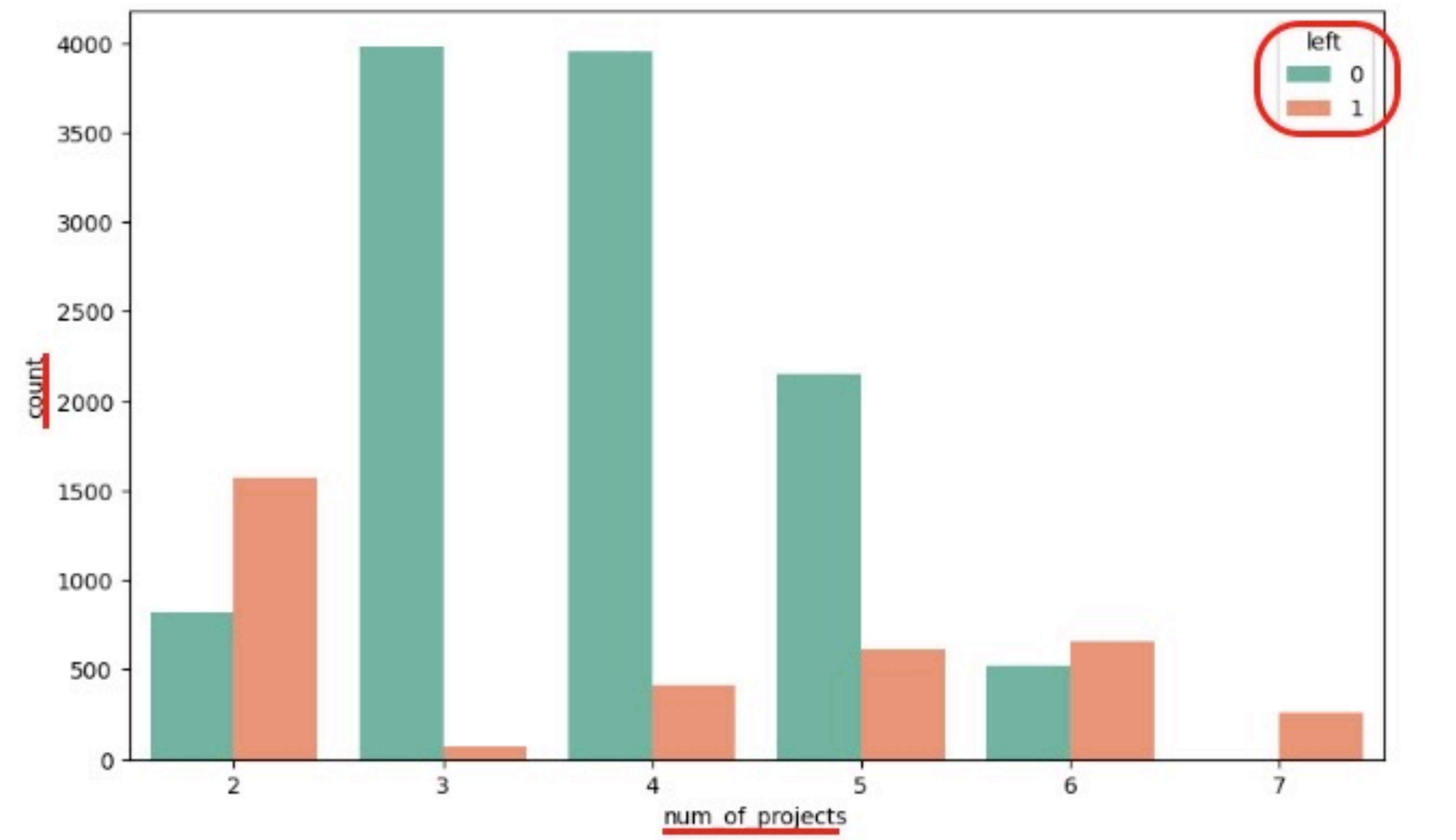
EDA

Analyze df column by column

num_of_projects

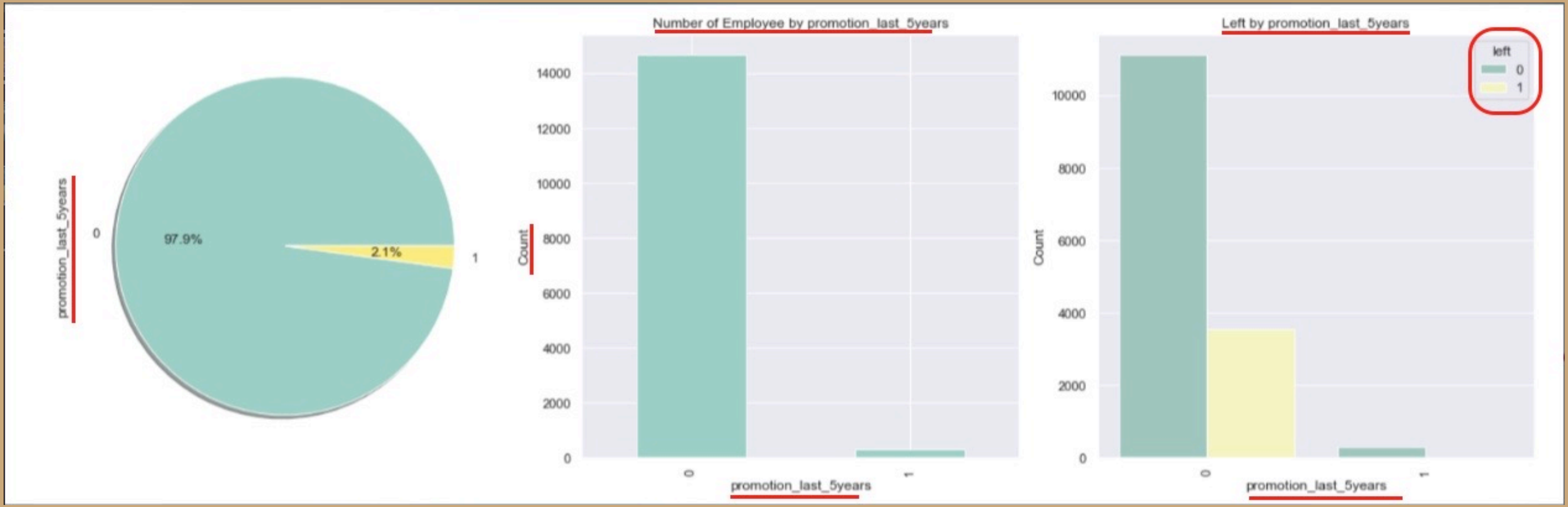
In [237…
```
df.num_of_projects.value_counts()
```
Out[237…
```
4    4365
3    4055
5    2761
2    2388
6    1174
7     256
Name: num_of_projects, dtype: int64
```
In [238…
```
df.num_of_projects.value_counts(normalize=True)
```
Out[238…
```
4    0.291
3    0.270
5    0.184
2    0.159
6    0.078
7    0.017
Name: num_of_projects, dtype: float64
```
In [239…
```
df.num_of_projects.describe()
```
Out[239…
```
count   14999.000
mean        3.803
std         1.233
min         2.000
25%         3.000
50%         4.000
75%         5.000
max         7.000
Name: num_of_projects, dtype: float64
```

```
sns.countplot(x=df.num_of_projects, palette="Set2");
```


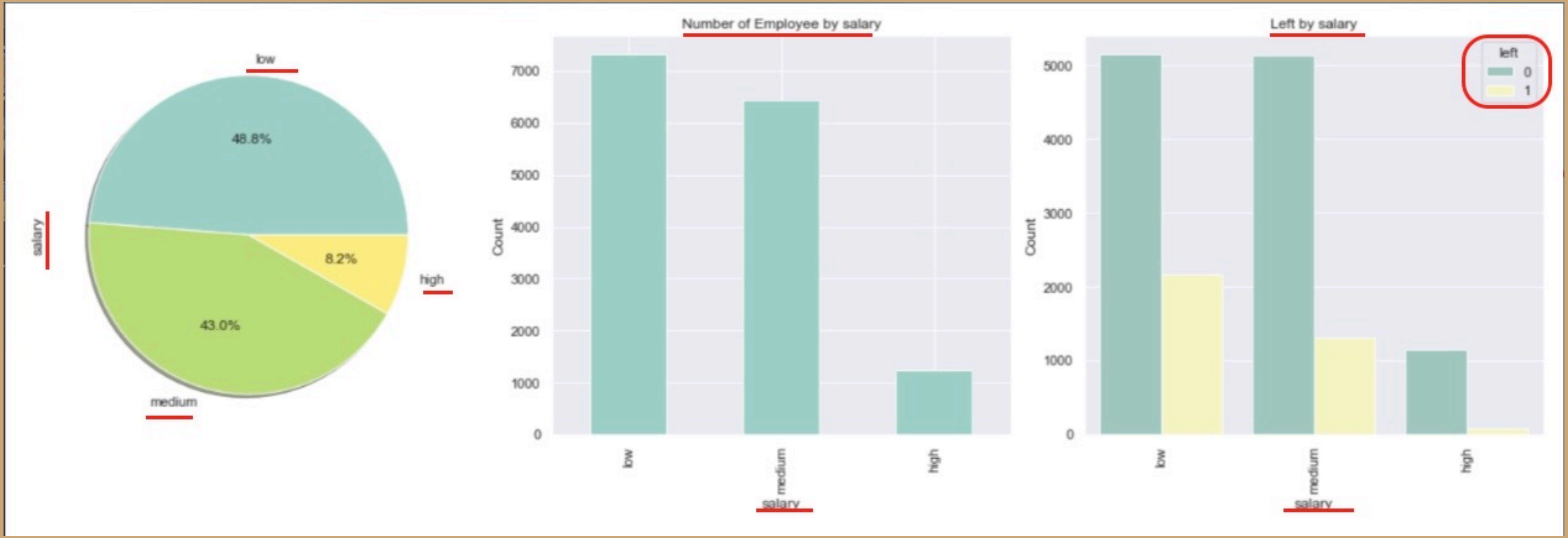

```
sns.countplot(x="num_of_projects", hue="left", palette="Set2", data=df);
```


Data Visualization

1.

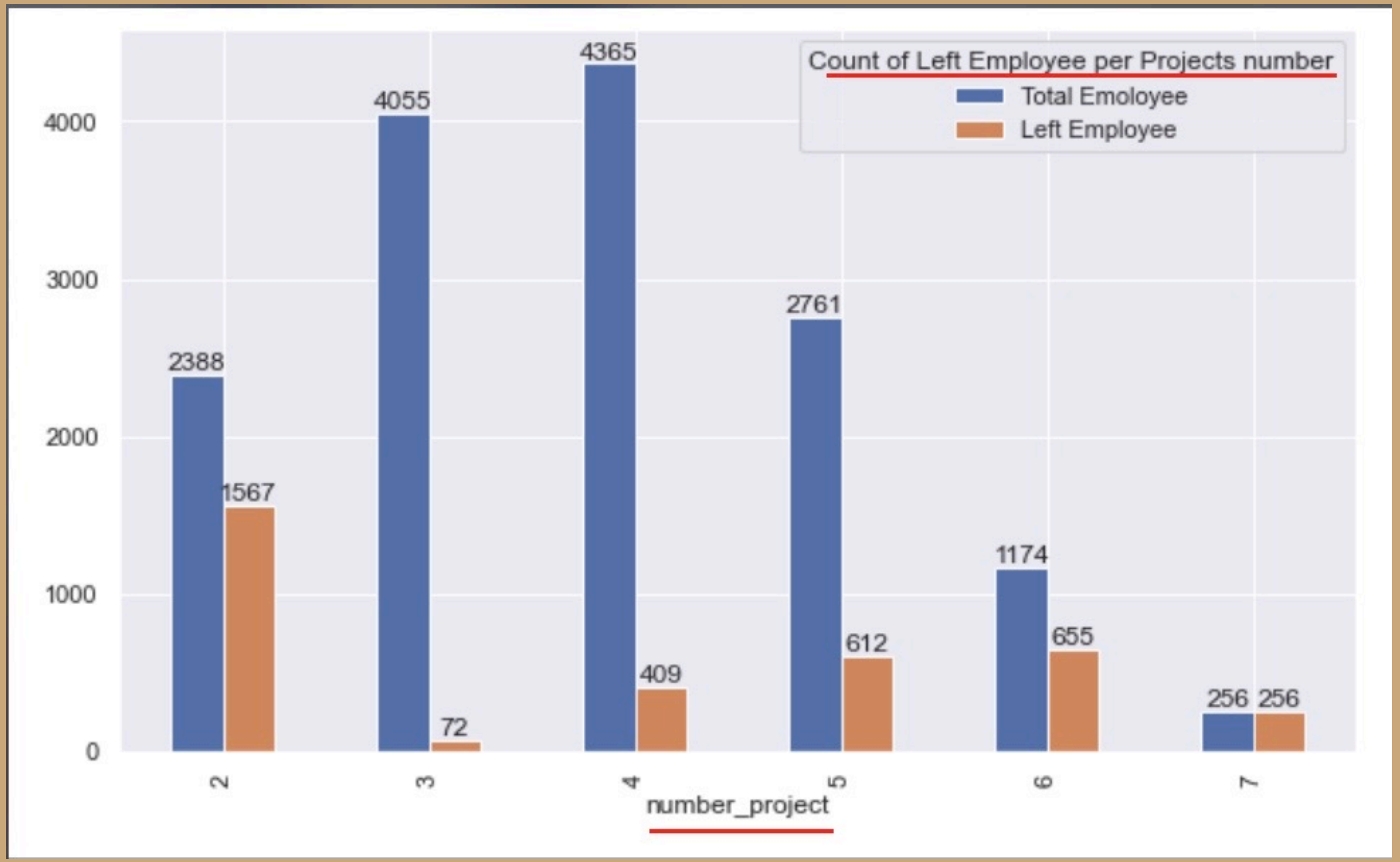


2.

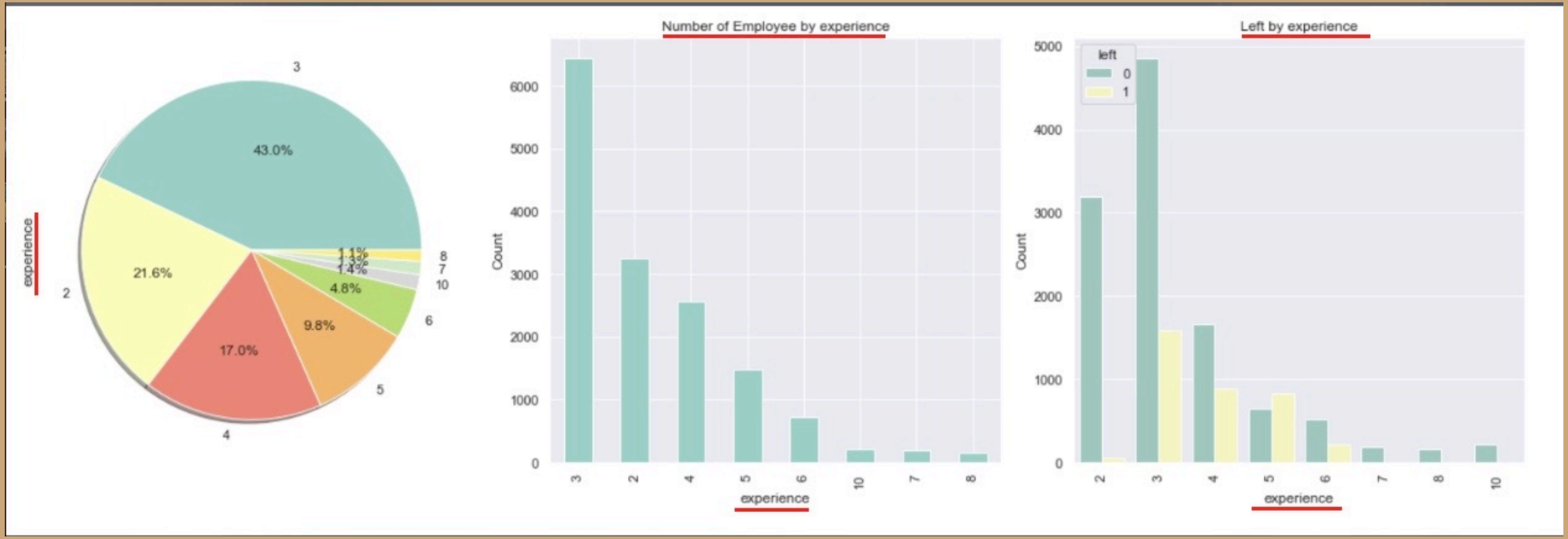


Data Visualization

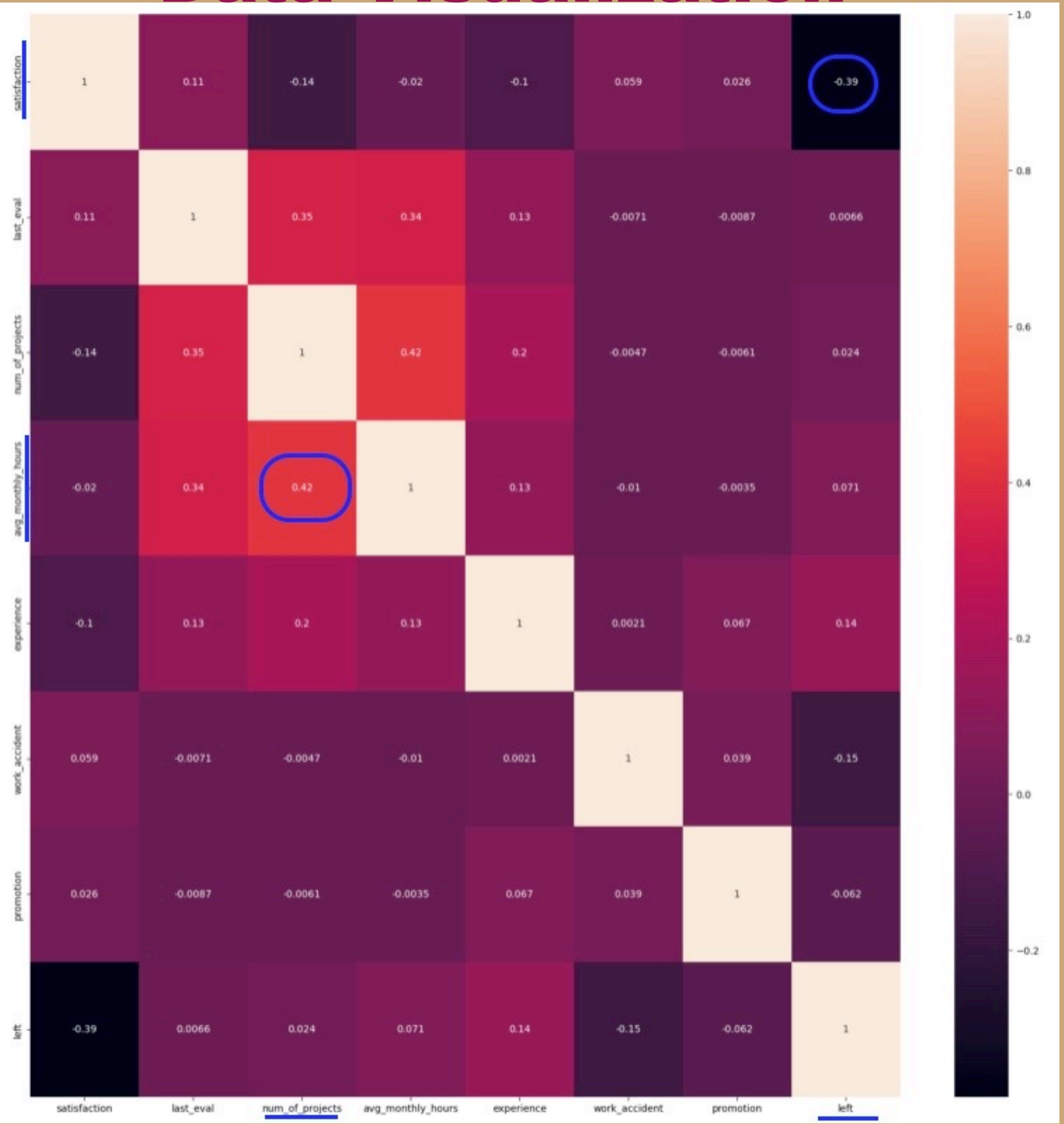
1.



2.



Data Visualization



Data Pre-Processing

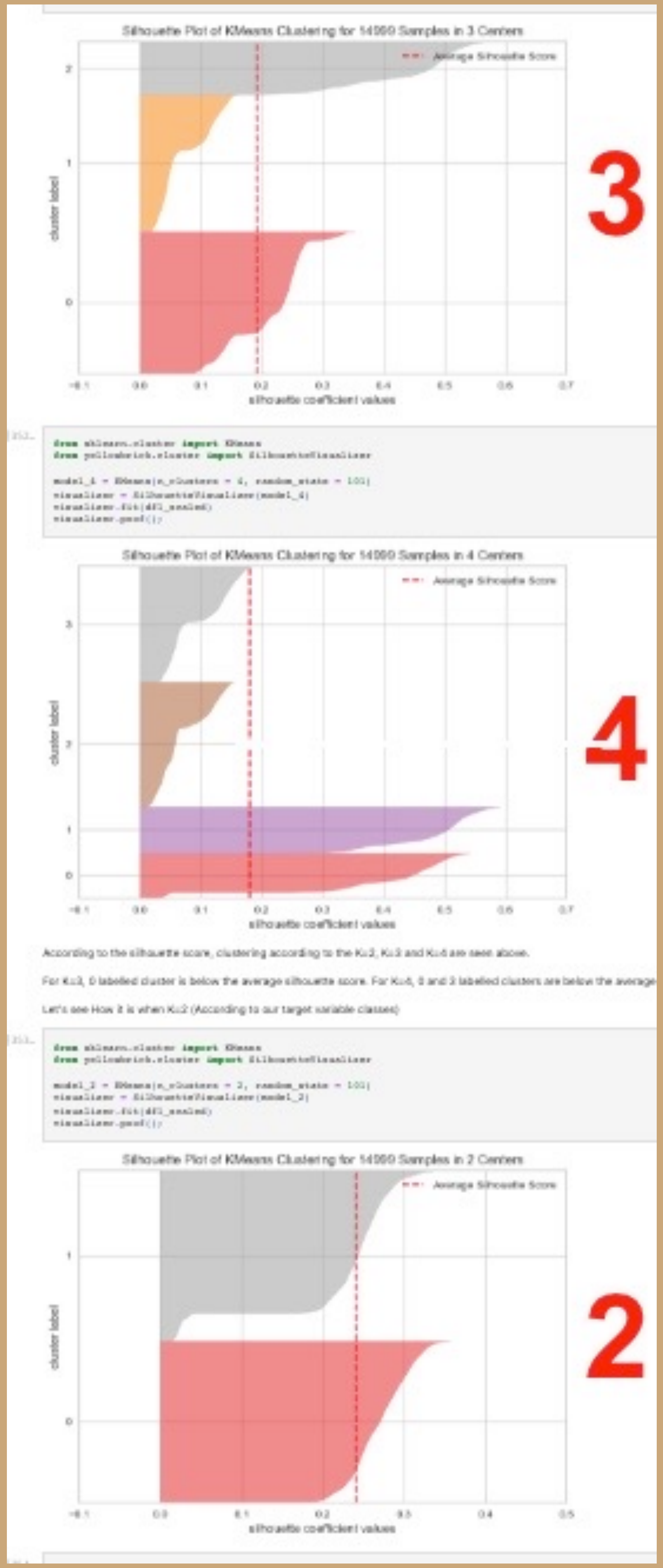
```
column_trans = make_column_transformer(
    (OneHotEncoder(handle_unknown="ignore", sparse=False), ["department"]),
    (OrdinalEncoder(handle_unknown='use_encoded_value', unknown_value=-1), ["salary"]),
    remainder=MinMaxScaler()
)
X = pd.DataFrame(data=column_trans.fit_transform(X), columns=column_trans.get_feature_names_out())
```

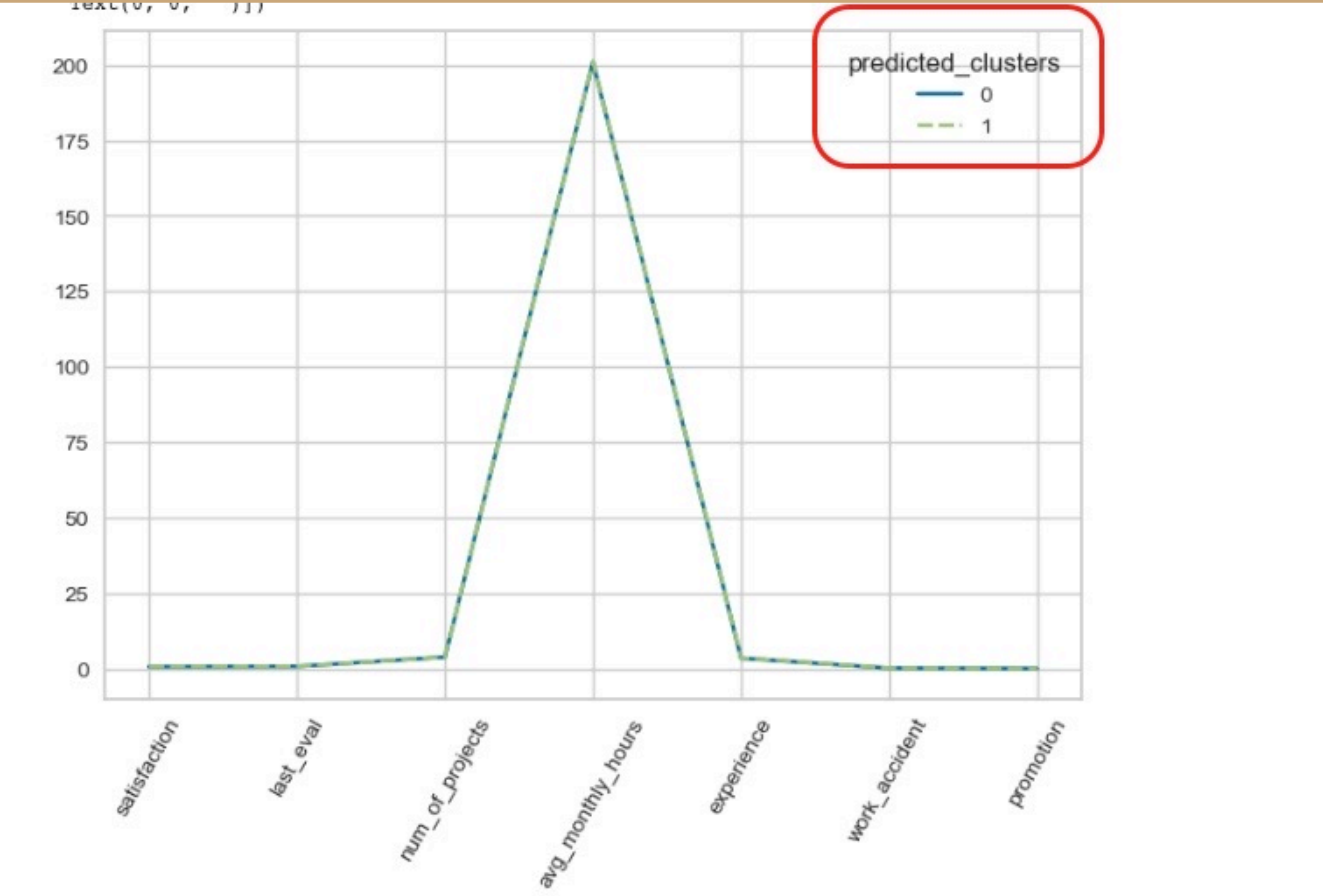
OneHotEncoder [“department”]

OrdinalEncoder [“salary”]

MinMaxScaler —> Remainder

Cluster Analysis





As seen above, the columns in the data set do not separate from each other. All columns are intertwined with each other.

As seen above it is visually obvious that clustering is not a good approach to our dataset.

From now on we are going to use classification models to make churn predictions.

Model Building

Logistic Regression

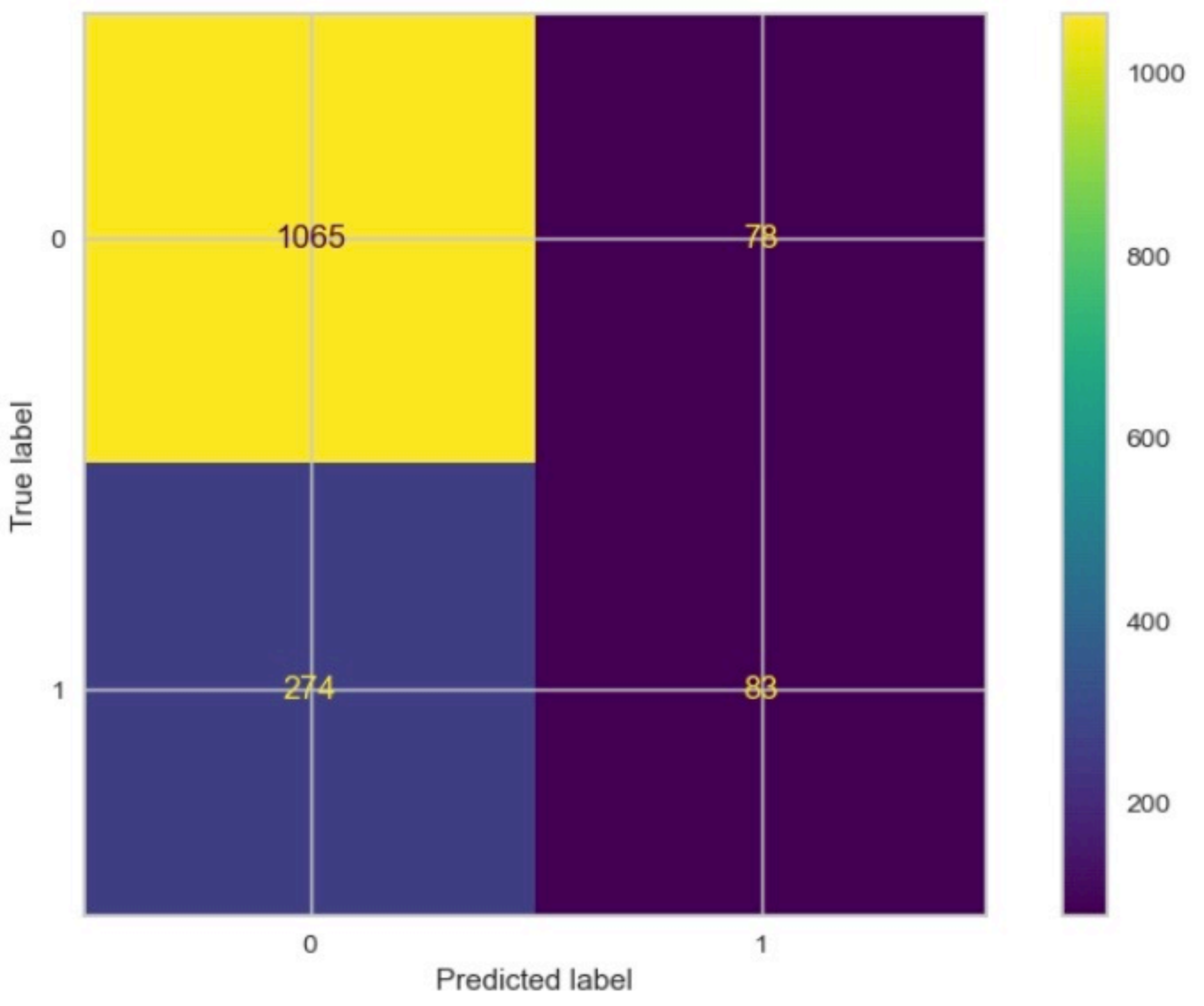
[565…

```
eval_metric(grid_model, X_train, y_train, X_test, y_test)
```

```
Test_Set

[[873 270]
 [ 75 282]]
              precision    recall  f1-score   support

           0       0.92      0.76      0.84      1143
           1       0.51      0.79      0.62       357

    accuracy                           0.77      1500
   macro avg       0.72      0.78      0.73      1500
weighted avg       0.82      0.77      0.78      1500


Train_Set

[[7616 2669]
 [ 693 2521]]
              precision    recall  f1-score   support

           0       0.92      0.74      0.82     10285
           1       0.49      0.78      0.60      3214

    accuracy                           0.75     13499
   macro avg       0.70      0.76      0.71     13499
weighted avg       0.81      0.75      0.77     13499
```

[551…

```
plot_confusion_matrix(log_model, X_test, y_test);
```



Model Building

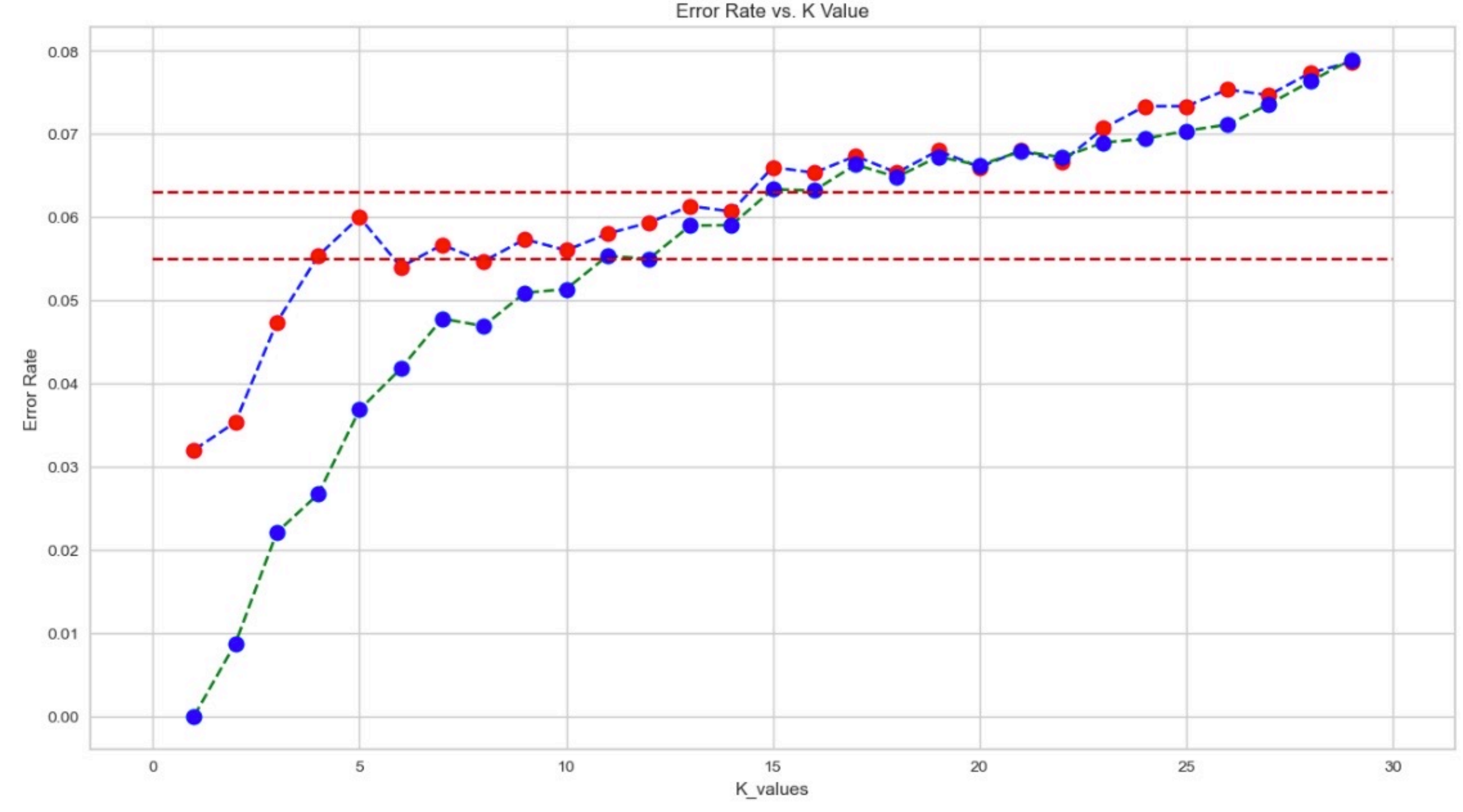
KNN

Test Set

```
[[1101   42]
 [  15  342]]
              precision    recall  f1-score   support

           0       0.99      0.96      0.97      1143
           1       0.89      0.96      0.92       357

    accuracy                           0.96      1500
   macro avg       0.94      0.96      0.95      1500
weighted avg       0.96      0.96      0.96      1500
```

Train_Set

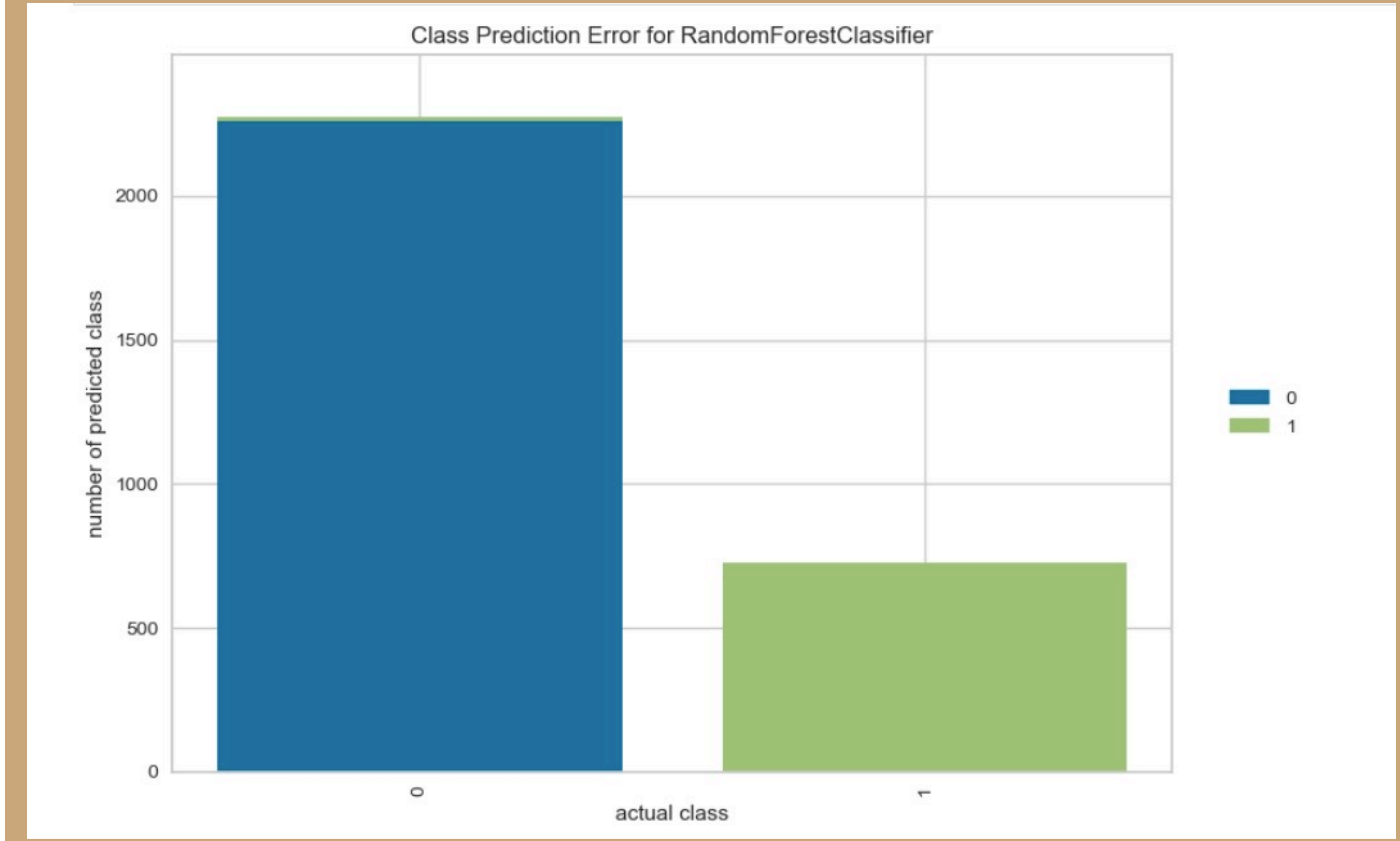
```
[[10285     0]
 [    0  3214]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10285
           1       1.00      1.00      1.00      3214

    accuracy                           1.00     13499
   macro avg       1.00      1.00      1.00     13499
weighted avg       1.00      1.00      1.00     13499
```



Model Building

RainForest

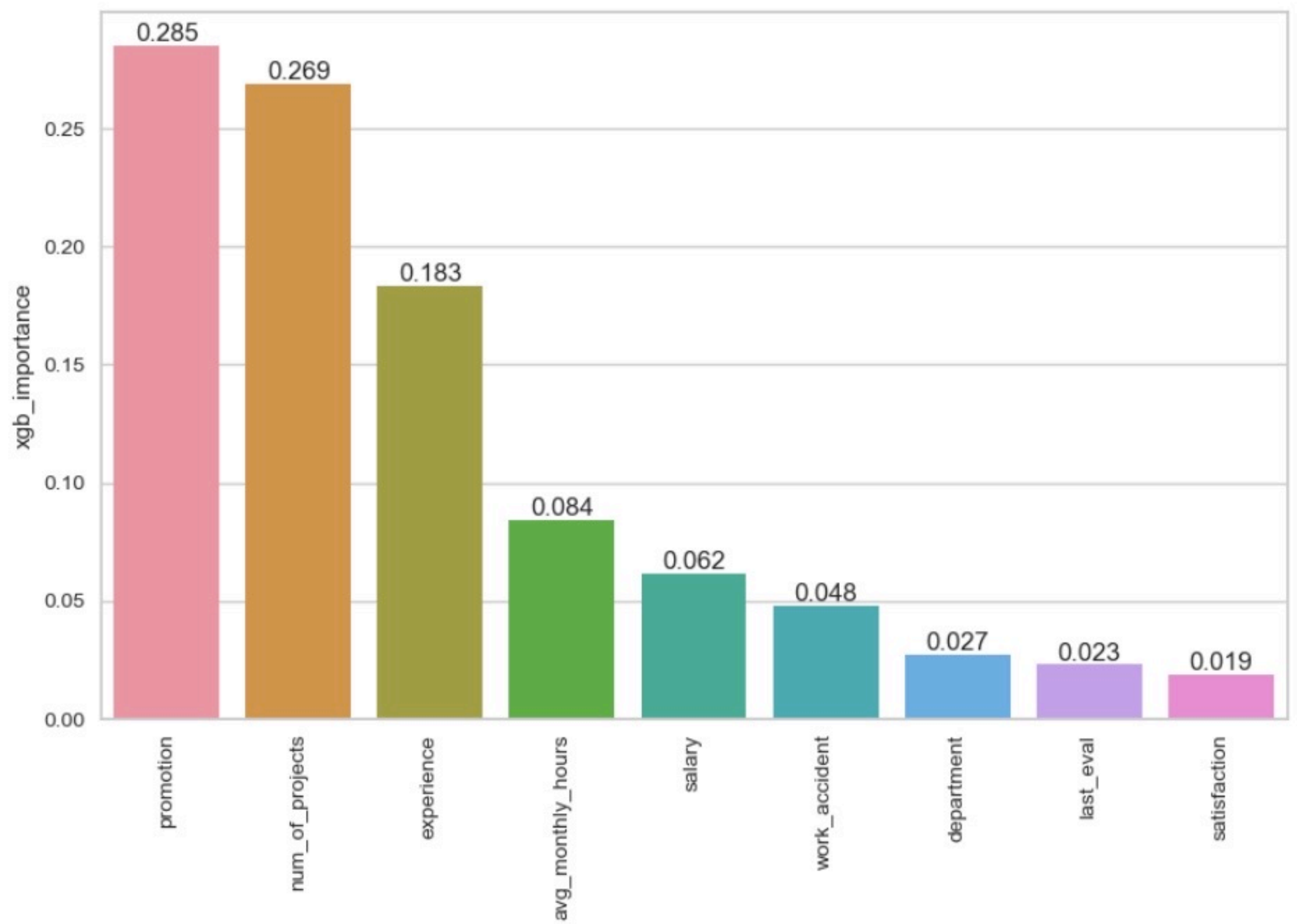
```
RF_tuned
------------------
Test Set
[[2239   23]
 [  38  700]]
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      2262
           1       0.97      0.95      0.96       738

    accuracy                           0.98      3000
   macro avg       0.98      0.97      0.97      3000
weighted avg       0.98      0.98      0.98      3000

Train Set
[[9098   68]
 [ 137 2696]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      9166
           1       0.98      0.95      0.96      2833

    accuracy                           0.98     11999
   macro avg       0.98      0.97      0.98     11999
weighted avg       0.98      0.98      0.98     11999
```



Model Building

XGBoost

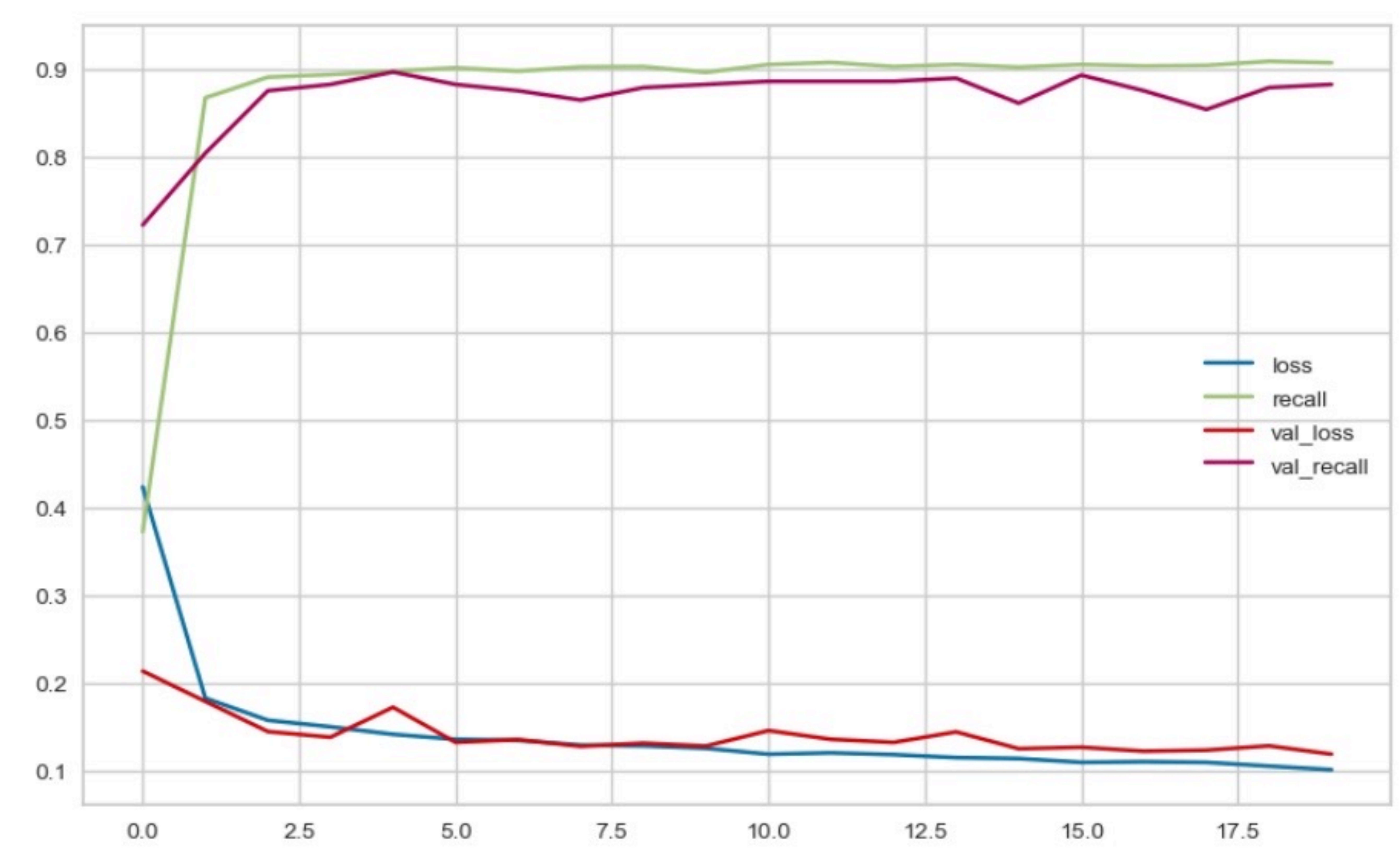
```
Test_Set
[[2257    5]
 [  15  723]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      2262
           1       0.99      0.98      0.99       738

    accuracy                           0.99      3000
   macro avg       0.99      0.99      0.99      3000
weighted avg       0.99      0.99      0.99      3000

Train_Set
[[9166    0]
 [   0 2833]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9166
           1       1.00      1.00      1.00      2833

    accuracy                           1.00     11999
   macro avg       1.00      1.00      1.00     11999
weighted avg       1.00      1.00      1.00     11999
```



Model Building

ANN

```
47/47 [==============================] - 0s 1ms/step
[[1098   45]
 [  28  329]]
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1143
           1       0.88      0.92      0.90       357

    accuracy                           0.95      1500
   macro avg       0.93      0.94      0.93      1500
weighted avg       0.95      0.95      0.95      1500
```



Model Comparing

Model Deployment

Select Employee's Attributes

Satisfaction

0,09

Last Evaluation

0,36

Number of Projects

1

1 7

Average Monthly Hours

1

1 310

Experience

1

1 10

Work Accident

Yes

Promotion

Yes

Department

sales

Salary

low

Employee Churn Prediction App

Whether your employees will continue to work with you or not ? Let's See !

Please fill the attributes on the left hand side to make run the model properly.

	satisfaction	last_eval	num_of_projects	avg_monthly_hours	work_accident	experience	promotion
0	0.0900	0.3600	1	1	1	1	1

Check the features you selected from the table above. If correct, press the Predict button.

Predict

Your employee will ** leave.**

Any question/feedback?

Thank you!