Machine Learning Project 16 Jun 2025

Laurea Magistrale in Telecommunications engineering
Università di Bologna

Name: Giorgio Cascelli
Course: Optimization and Machine Learning M

Project objective

The objective of this project is to develop a machine learning model that can predict the salaries for data scientist based on relevant features such as experience level, company location, and other factors using the "ds_salaries" dataset.

1 Data Exploration and Preprocessing:

1.a Load the "ds_salaries" dataset

First thing, we import the dataset "ds_salaries":

```
df = pd.read_csv('ds_salaries.csv')
```

Next, we ensure numerical precision by converting all numeric columns to the float64 data type, which is fully supported by the libraries used:

```
for col in df.select_dtypes(include=['number']).columns:
    df[col] = df[col].astype('float64')
```

1.b Handle any missing values or outliers

We first identify duplicate rows:

```
duplicate_rows = df[df.duplicated()]
```

A total of 1171 duplicates are found. While duplication could reflect real-world overlap (e.g., identical roles across companies), retaining them risks overfitting. Thus, all duplicates are removed, reducing the dataset to 2584 entries:

```
df = df.drop_duplicates()
```

We check for missing values:

```
nan_entries = df[df.isna().any(axis=1)]
```

No missing entries are found.

To detect outliers, we visualize the salary distribution (Figure 1). Although roughly normal, several high end outliers are present. We filter out salaries exceeding $400,000 to mitigate their influence:

```
df = df[df['salary_in_usd'] <= 400000]
```

This reduces the dataset to 2577 entries.

1.c Visualize the distribution of the "job_title" and "employee_residence"

Figure 2 reveals a large variety of job titles. We retain only entries where the **last word** of the job title belongs to the 10 most frequent values, see Figure 3, reducing the dataset to 2547 entries.

Regarding employee residence, U.S. entries are significantly overrepresented, potentially biasing the model toward U.S. salaries. To improve balance, we randomly retain one in every nine U.S. entries. This decision, determined via experimental tuning, improves generalization, see Figure 3. The final dataset contains 879 entries.

Although significantly smaller, this version promotes diversity. Retaining only U.S. entries would yield 1876 samples but with diminished global relevance.

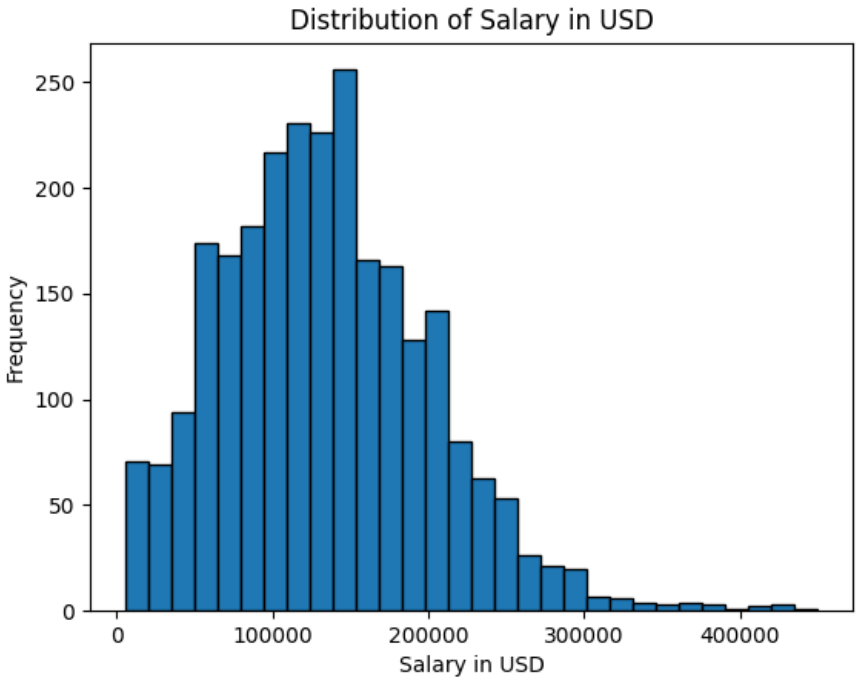

Figure 1: Histogram of the distribution of the Salaries in USD

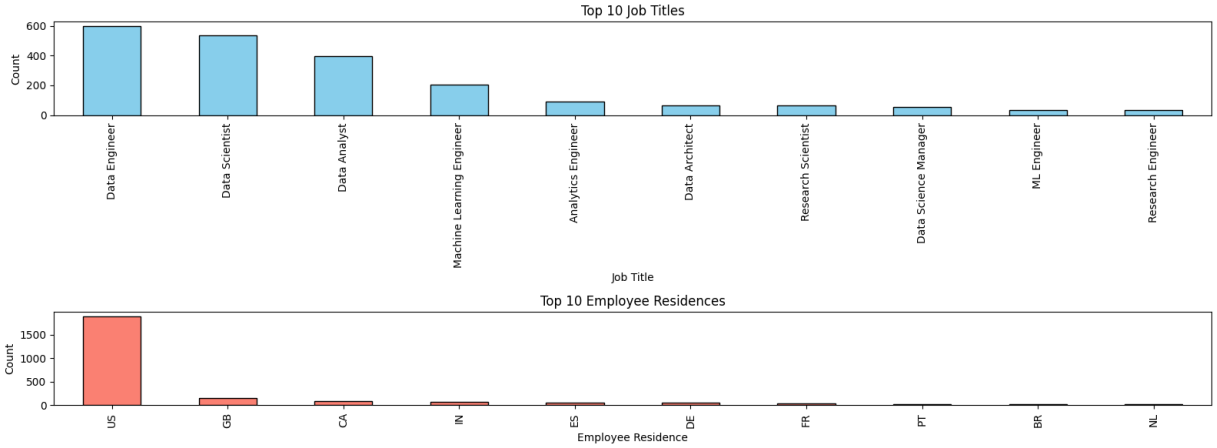

Figure 2: Distribution of top 10 Job Titles and Employee Residences.

2 Feature Selection and Engineering

2.a Identify the relevant features from the dataset that can potentially influence salary prediction

Feature evaluation highlights the following:

- The **work year** would be relevant, but entries are only from 2020 to 2023, so it may not add much value and just add more complexity.
- The **experience level** is a crucial factor, as it typically correlates with salary.
- The **employment type** would be relevant but most of the entries are 'full-time', so it may not add much value.
- The **job title** is important as it directly relates to the role and responsibilities, which influence salary.
- The **employee residence** and **company location** can affect salary due to cost of living differences.

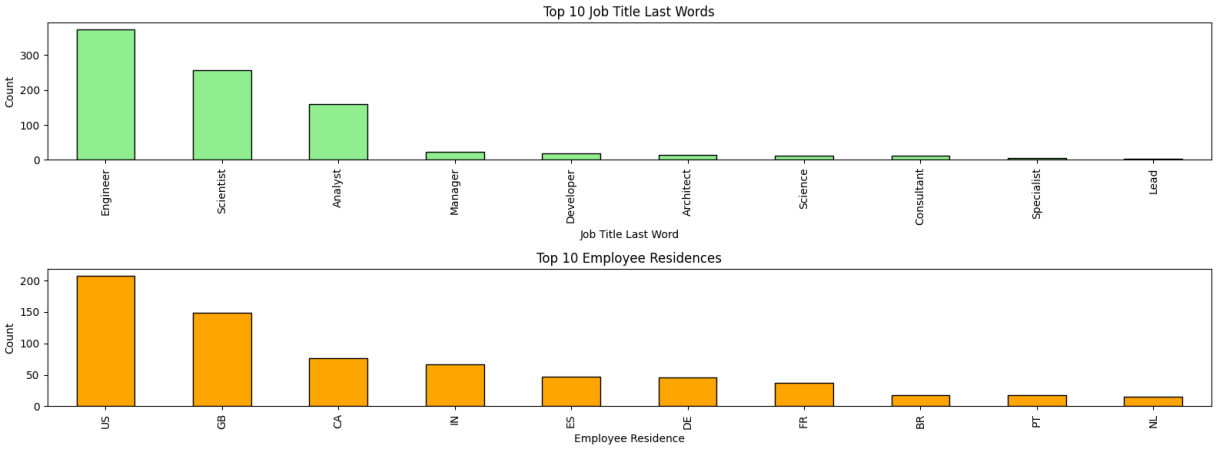

Figure 3: Distribution of top 10 Job Titles last word and Employee Residences after keeping one every nine US residents.

- The **company size** can also influence salary, as larger companies may have more resources to pay higher salaries.
- The **remote ratio** does not seem to have an impact on salary based on the dataset. In fact considering the average salary of different remote ratios, it appears that remote work does not significantly affect salary, see Fig. 4.

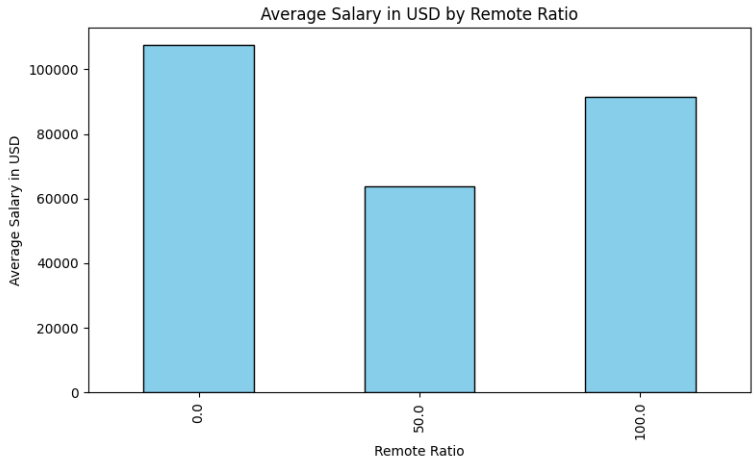

Figure 4: Average salary, by remote ratio

In the end, the relevant features are: 'experience_level', job_title', 'employee_residence', 'company_location', 'company_size'.

2.b Perform feature engineering, scaling features

It is better to work with normalized values of US is salary:

```
df['salary_in_usd_normalized'] = df['salary_in_usd'] / df['salary_in_usd'].max()
```

Categorical features were then encoded as numerical values to enable their use in the machine learning model. One-hot encoding was initially considered; however, due to the high cardinality of several features (e.g., job title, employee residence), it resulted in a substantial increase in feature dimensionality, which ultimately reduced the model's effectiveness on this relatively small dataset.

Instead, a more compact encoding strategy was adopted. For ordinal features such as `experience_level` and `company_size`, a manual mapping was applied to preserve their intrinsic ordering. This is crucial, as these variables carry a natural progression that is informative for the model.

```
experience_map = { 'EX':4,'SE':3, 'MI':2, 'EN':1}
size_map = {'S': 1, 'M': 2, 'L': 3}
```

For the features `job_title`, `employee_residence`, and `company_location`, automatic encoding using categorical codes was applied, as these are nominal variables with no inherent order.

```
for col in ['job_title', 'employee_residence', 'company_location']:
    df[col] = pd.Categorical(df[col]).codes
```

3 Model Development and Evaluation

3.a Split the dataset into training and testing subsets

The dataset was randomly divided into a training and a testing set, with an 80% 20% split. A fixed random seed was used to ensure reproducibility of results. The normalized salary in USD was used as the target variable, and the relevant features identified during preprocessing served as model inputs.

```
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
X_train = train_df[relevant_features].copy()
X_test = test_df[relevant_features].copy()
y_train = train_df['salary_in_usd_normalized']
y_test = test_df['salary_in_usd_normalized']
```

3.b Choose an appropriate machine learning algorithm

Three regression models were evaluated:

- **Linear Regression**: A baseline model that assumes a linear relationship between the input features and the target.
- **Histogram-based Gradient Boosting (HGB)**: An efficient implementation of gradient-boosted decision trees optimized for large datasets.
- **Random Forest Regression**: An ensemble of decision trees trained with bootstrap aggregation, offering strong performance with minimal tuning.

While Linear Regression produced relatively poor performance due to the non-linearity and categorical nature of the features, both ensemble models achieved significantly better results. The Random Forest Regressor outperformed the others slightly in terms of accuracy, as shown in Table 1.

The rationale behind these results, as well as the interpretation of the evaluation metrics, is discussed in the following sections.

x	Linear	Histogram	Random Forest
MSE	0.02	0.01	0.01
R^2	0.34	0.60	0.64

Table 1: Brief recap of results from different tested models

3.c Train the model using the training data and evaluate its performance using suitable evaluation metrics

To find the best performance of the Random Forest model, a randomized grid search was conducted to explore combinations of key hyperparameters. The search was guided by R^2 as the scoring metric and was performed using three-fold cross-validation. The grid included parameters such as the number of estimators, tree depth, minimum samples per leaf and split, the criterion for node splitting, and the feature sampling strategy.

```
param_grid = {
    'criterion': ['squared_error', 'absolute_error'],
    'n_estimators': [300, 400, 500, 600],
    'max_depth': [20, 30, 40, 50, None],
    'min_samples_split': [2, 5, 8, 12],
```

```
    'min_samples_leaf': [1, 2, 3, 5],
    'max_features': ['sqrt', 'log2', 0.7, 0.8] }
search = RandomizedSearchCV(
    RandomForestRegressor(random_state=42),
    param_distributions=param_grid,
    n_iter=40,
    scoring='r2',
    cv=3,
    n_jobs=-1,
    verbose=1,
    random_state=42)
```

The optimal hyperparameter configuration returned by the search is reported in Table 2.

Parameter	Value
n_estimators	300
min_samples_split	12
min_samples_leaf	2
max_features	log2
max_depth	50
criterion	squared_error

Table 2: Best hyperparameters found by the randomized search

The model was evaluated using two primary metrics: the Mean Squared Error (MSE) and the coefficient of determination (R^2).

- **MSE** quantifies the average squared difference between the predicted and actual salaries. A lower MSE indicates better predictive accuracy. The final model yielded an MSE of 0.01 (on normalized salaries), corresponding to an average error of approximately 10% when rescaled to the original salary range.
- **R^2 Score** measures the proportion of variance in the target variable that is explained by the model. The best Random Forest model achieved $R^2 = 0.63$, which was later improved to $R^2 = 0.64$ after manual tuning. This level of performance is considered good for real-world salary prediction tasks, which often involve substantial unobserved factors.

3.d Fine-tune the model, by adjusting hyperparameters to improve performance

Following the randomized search, further manual tuning of the Random Forest model yielded slight improvements in performance. The manually optimized hyperparameters are listed in Table 3.

Parameter	Value
n_estimators	200
min_samples_split	5
min_samples_leaf	1
max_features	sqrt
max_depth	None
criterion	squared_error

Table 3: Best hyperparameters found by manual fine tuning

4 Prediction and Interpretation

After training and optimizing the model, predictions were generated on the test set and compared against the actual normalized salary values. Figure 5 shows the distribution of predicted salaries versus the true salaries in the test set. The overall shape and central tendency of the two distributions are similar, indicating that the model has captured the key trends in the data. However, some deviations at the distribution tails suggest the model may still underperform for extremely high or low salaries, likely due to limited representation of such cases in the training data.

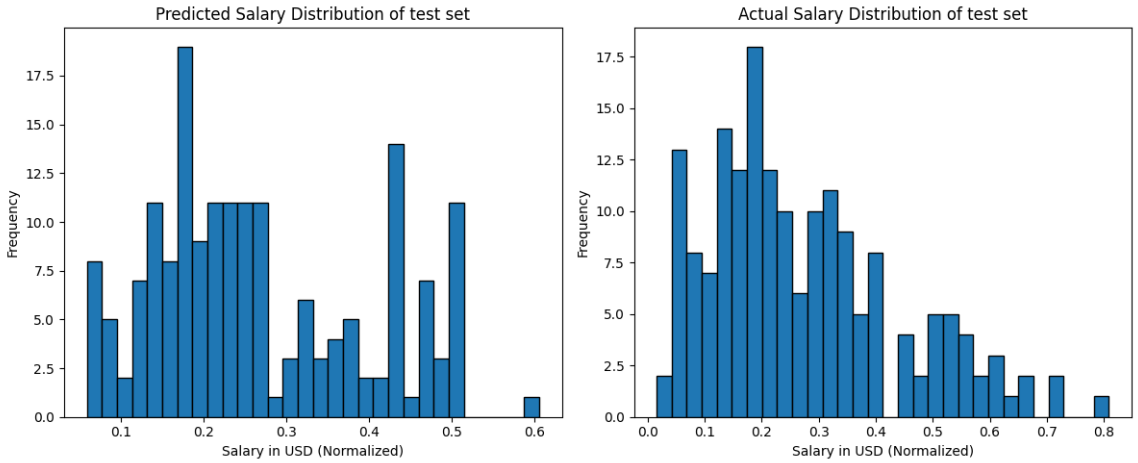

Figure 5: Predicted vs Actual salary for test data

To gain insight into which input features most influenced the model's predictions, we computed feature importance based on the trained Random Forest model. As shown in Figure 6, the three most impactful variables were:

- **Employee Residence**: Geographic location affects salary expectations due to regional differences in cost of living and labor market conditions.
- **Job Title**: Job roles directly reflect responsibilities and required expertise, which naturally influences compensation.
- **Experience Level**: This ordinal feature has a strong and expected correlation with salary, as more experienced professionals tend to receive higher pay.

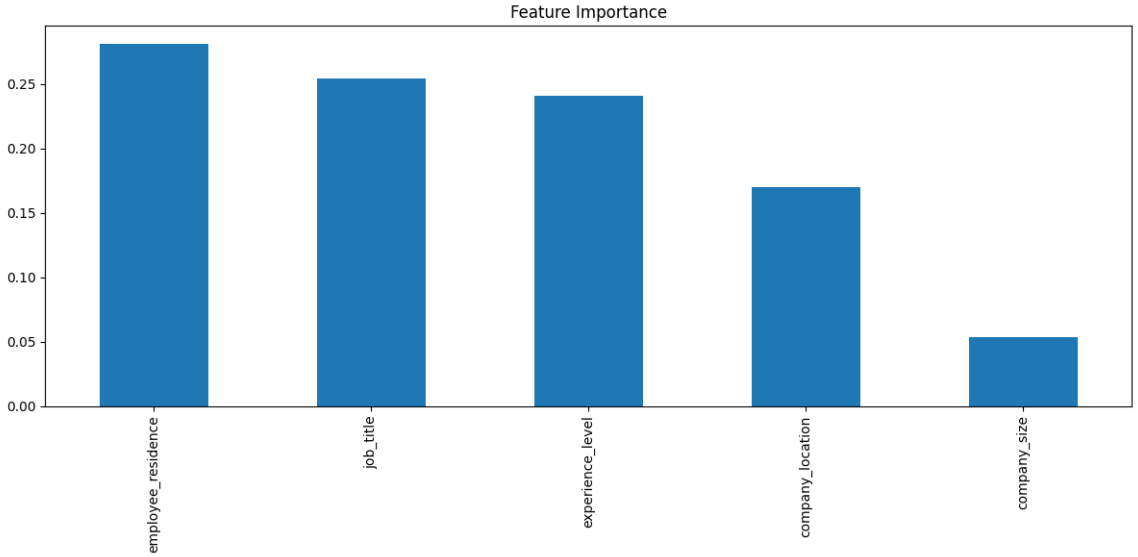

Figure 6: Feature importance

Other features such as company size and company location also contributed to predictions, although with less influence. Interestingly, the relatively low importance of company location compared to employee residence may reflect a shift toward remote work dynamics in the data science industry, where salaries are increasingly decoupled from physical office locations.

To assess whether the model has overfitted the training data, we evaluate its performance on the same data used for training. Figure 7 shows the comparison between predicted and actual normalized salaries on the training set. A near-perfect alignment would indicate potential overfitting, whereas consistent deviations could suggest underfitting or model limitations.

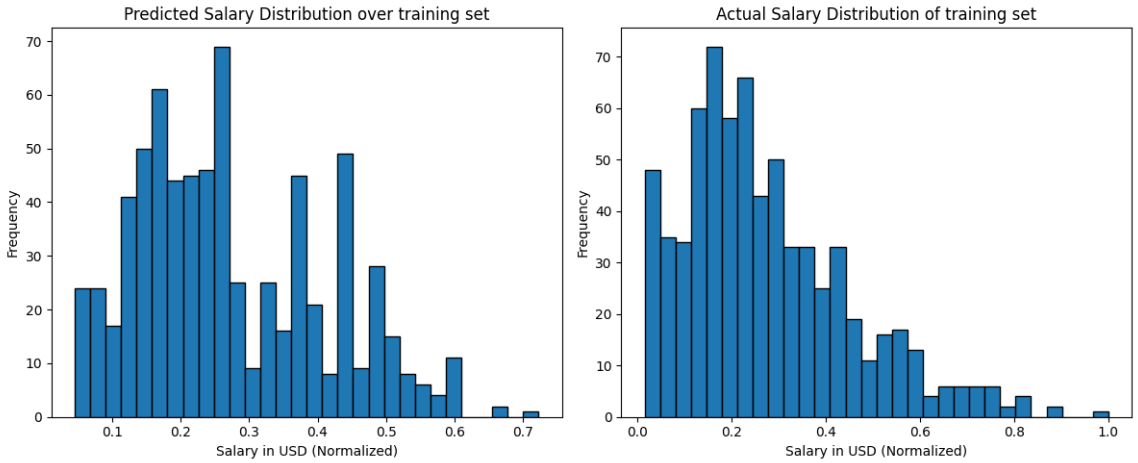

Figure 7: Predicted vs Actual salary for training data

In addition, we analyze the residuals, which is the difference between the predicted and actual salaries, to assess how well the model captures the underlying structure of the data. The residual distribution is shown in Figure 8. Ideally, residuals should be normally distributed with zero mean and minimal variance, which would indicate that the model has extracted all meaningful patterns and the remaining error is unstructured noise.

In our case, the residuals follow a roughly Gaussian distribution with a mean close to zero, although the distribution is not sharply peaked. This suggests that while the model has learned the general structure of the data, some variance remains unexplained, likely due to noise or features not included in the dataset.

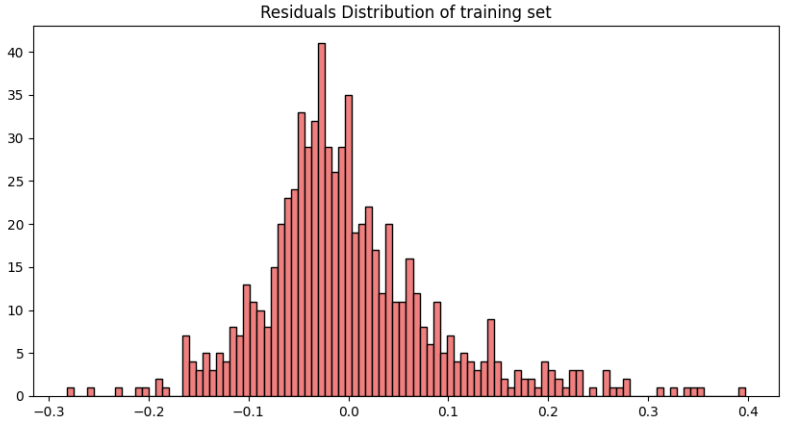

Figure 8: Residual distribution

5 Conclusion and Recommendations

This project aimed to build a machine learning model to predict data scientist salaries using the ds_salaries dataset. Through a careful process of data cleaning, feature selection, encoding, and model tuning, the final Random Forest Regressor achieved a Mean Squared Error (MSE) of 0.01 and an R^2 score of 0.64 on the test set.

These results indicate that the model is able to explain approximately 64% of the variance in salaries. In the context of real-world salary prediction, where numerous external and unquantified variables (e.g., individual negotiation, company-specific policies, and macroeconomic factors) come into play, an R^2 score above 0.60 is considered a solid performance.

The most influential features identified were employee residence, job title, and experience level. This aligns with expectations, as geographic location affects cost of living and regional salary standards; job title reflects role responsibilities; and experience level is closely tied to compensation.

Due to significant class imbalance, particularly the over representation of U.S. based entries, the dataset was down sampled to retain a single U.S. sample out of every nine. This strategy, while reducing the total sample size to 879 entries, improved the global balance of the dataset and likely contributed to better generalization. The alternative approach (i.e., keeping only U.S. entries) would have yielded a larger dataset but at the cost of international representativeness.

The project also explored various encoding strategies. One-hot encoding was initially considered but discarded in favor of ordinal and categorical encoding due to the relatively small dataset and high cardinality of some features (e.g., job titles and locations). Importantly, for ordered features such as `experience_level` and `company_size`, ordinal encoding was deliberately applied to preserve the inherent ranking. This choice allowed the model to better exploit the structural relationships embedded in these features.

Recommendations

- **Dataset Expansion:** Collecting more balanced and international data could improve the model's generalizability.
- **Feature Enrichment:** Including additional features such as industry sector, education level, or job-specific skills might capture more variance in salaries.
- **Model Ensemble:** While Random Forest performed best in this study, combining it with other models in an ensemble could potentially yield improved results.
- **Feature Compression and Embedding:** Another promising direction would be to categorize and embed the relevant features more effectively, reducing the input space by assigning a numerical value to each full input instance based on all features. This would transform the model's task into learning a mapping from a single high-cardinality numerical input to the output. Finding an optimal encoding or embedding strategy for this compressed input is itself a machine learning problem, possibly solvable via neural networks.

In summary, the project demonstrates that machine learning models can provide valuable salary estimation based on job-related features, though real-world applications should account for limitations in the data and consider continual retraining with updated information.