

GAZI UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING



CENG313 – INTRODUCTION TO DATA SCIENCE
PROF. DR. ŞEREF SAĞIROĞLU

AI-BASED SALARY ESTIMATION AND RECOMMENDATION SYSTEM
[\(GitHub\)](#)

PREPARED BY GROUP 15

Ahmet Emir Coşkun - 22118080049

Enes İnançlı - 23118080051

Mustafa Temur Turan - 22118080021

December 8, 2025

1. Introduction

The purpose of this project is to develop an **AI-based salary estimation and recommendation system** that predicts salary ranges for employees and provides data-driven insights for individuals and employers. The system uses demographic, educational, and career-related information—such as age, experience, education level, job title, and seniority—to estimate an expected salary using machine learning techniques.

The project has two main goals:

1. **For Individuals:**
To help users understand market salary expectations based on their qualifications.
2. **For Employers:**
To allow companies to upload internal employee data, analyze salary structures, and generate fair salary recommendations for new hires.

This project follows a complete **end-to-end data science pipeline**, beginning with data collection and preprocessing, followed by model development, evaluation, and deployment through an interactive UI built with Streamlit.

The project evolved throughout the semester. Several methodological choices changed as the team discovered new insights—for example, transitioning from simple label encoding to **target encoding**, restructuring job title normalization, and substantial refinement of the dataset due to quality issues. All changes were documented in the project's internal reports and are reflected in this final submission.

2. Dataset

2.1 Dataset Sources

Two open-source datasets were used for the initial phase of the project:

- [Salary Data.csv](#)
- [Salary by Job Title and Country.csv](#)

These datasets are documented in detail inside:

- *data/raw/README.md*

Combined, they originally contained **over 13,000 records**. However, after extensive cleaning, filtering, and deduplication, the final modeling dataset contained **approximately 1,500 high-quality observations**.

This reduction was expected, as both the Milestone Report and Data Processing Report documented severe inconsistencies such as:

- non-standard job titles,
- missing or incorrect salary values,
- mismatched country/currency information,
- unrealistic ages or experience levels,
- duplicate entries.

This aligns with real-world data science practice, where raw datasets often require heavy refinement before becoming modeling-ready.

2.2 Dataset Characteristics

Final dataset features:

Feature	Type	Description
age	Numeric	Age of employee (18–70 after filtering)
job_title	Categorical	Normalized job title after semantic and typo correction
education_level	Categorical (ordinal)	High School → PhD
years_experience	Numeric	Total years of professional experience
salary	Numeric	Annual salary (cleaned range: 20,000–200,000)
seniority_level	Categorical (binary)	Junior / Senior

Descriptive Statistics (from processed dataset)

- **Unique job titles:** 118
- **Mean salary:** ~\$112,905
- **Median salary:** ~\$110,000
- **Average age:** 35.27
- **Experience range:** 0–50 years

2.3 Dataset Screenshot Placement

1	job_title	education_level	years_experience	salary	seniority_level	
2	software engineer	bachelor	5.0	90000.0	junior	
3	data analyst	master	3.0	65000.0	junior	
4	manager	phd	15.0	150000.0	senior	
5	sales associate	bachelor	7.0	60000.0	junior	
6	general director	master	20.0	200000.0	junior	
7	marketing analyst	bachelor	2.0	55000.0	junior	
8	product manager	master	12.0	120000.0	junior	
9	sales manager	bachelor	4.0	80000.0	junior	
10	marketing coordinator	bachelor	1.0	45000.0	junior	
11	research scientist	phd	10.0	110000.0	senior	
12	software developer	master	3.0	75000.0	junior	
13	hr manager	bachelor	18.0	140000.0	junior	
14	financial analyst	bachelor	6.0	65000.0	junior	
15	project manager	master	14.0	130000.0	junior	
16	customer service representative	bachelor	2.0	40000.0	junior	
17	operations manager	bachelor	16.0	125000.0	junior	
18	marketing manager	master	7.0	90000.0	junior	
19	engineer	phd	12.0	115000.0	senior	
20	data entry clerk	bachelor	0.0	35000.0	junior	
21	sales director	bachelor	22.0	180000.0	junior	
22	business analyst	master	5.0	80000.0	junior	
23	vp of operations	master	19.0	190000.0	junior	
24	it support specialist	bachelor	2.0	50000.0	junior	
25	recruiter	bachelor	9.0	60000.0	junior	
26	financial manager	master	13.0	140000.0	junior	
27	social media specialist	bachelor	3.0	45000.0	junior	
28	software manager	master	11.0	110000.0	junior	
29	software developer	bachelor	1.0	40000.0	junior	
30	consultant	phd	15.0	140000.0	senior	

Figure 1 - Sample of the cleaned and processed dataset used for model training

3. Data Preprocessing

The data preprocessing pipeline was the most extensive part of the project. All steps were implemented in Python and documented in the `01_data_processing.md` report.

3.1 Schema Standardization

Columns across datasets were mapped into a **uniform schema**:

- age
- job_title
- education_level
- years_experience
- salary
- seniority_level (engineered)

3.2 Seniority Level Reconstruction

Seniority information was inconsistent. The team designed a hybrid decision rule:

- If dataset flag = senior → **senior**
- If job title contains “senior” → **senior**
- If job title contains “junior” → **junior**
- If neutral → **junior** (supported by median salary analysis)

This decision significantly improved feature consistency.

3.3 Job Title Normalization

Applied operations:

- Typo correction
- Unifying variations (“front end developer” → “frontend developer”)
- Semantic grouping (“developer” → “software developer”)
- Reducing unique titles from 125 → 118

```
# === JOB TITLE ADVANCED NORMALIZATION ===

# 1) Yazım hatası düzeltme
typo_map = {
    "juniour hr coordinator": "junior hr coordinator",
    "juniour hr generalist": "junior hr generalist",
    "social media man": "social media manager",
}

# 2) Benzer rolleri gruplayarak tek kategoriye düşürme
merge_map = {
    "customer service rep": "customer service representative",
    "customer success rep": "customer success manager",
    "developer": "software developer",
    "front end developer": "frontend developer",
    "back end developer": "backend developer",
    "full stack engineer": "fullstack engineer",
}

# 3) Daha tutarlı isimlendirme için ek standartizasyon
standardize_map = {
    "it support": "it support specialist",
    "it project manager": "project manager",
    "research director": "director of research",
    "director": "general director",
    "scientist": "research scientist"
}

# Tüm map'leri birleştir
jobtitle_map = {**typo_map, **merge_map, **standardize_map}

# Uygula
combined["job_title"] = combined["job_title"].replace(jobtitle_map)
```

Figure 2 - Python snippet showing the job title normalization pipeline used during data preprocessing.

3.4 Data Cleaning and Filtering

Actions taken:

- Removed salaries ≤ 0
- Removed unrealistic age / experience
- Removed top 1% outliers (cleaned upper bound: \$200,000)
- Converted text fields to numerical formats
- Median imputation for numeric columns
- “unknown” label for categorical missing values
- Duplicate removal (original: ~9,991 duplicates → final: 0)

3.5 Final Dataset Structure

A detailed multi-step preprocessing pipeline was implemented as part of the *01_data_preprocessing.ipynb notebook*. The pipeline includes dataset loading, merging, numeric and categorical cleaning, outlier removal, job title normalization, feature encoding, and final dataset export operations. Since the complete pipeline spans multiple cells and consists of more than a dozen sequential transformations, only the conceptual workflow is presented here. The full implementation can be found in the aforementioned notebook, where each step is documented and executed in a modular and reproducible manner.

4. Methods Used

4.1 Feature Engineering

Feature engineering was a crucial step in improving the model’s predictive performance and ensuring that both categorical and numerical attributes were represented in a meaningful, machine-interpretable format. In this project, two primary encoding strategies were used: **Ordinal Encoding** and **Target Encoding**.

1. Ordinal Encoding

Ordinal Encoding was applied to categorical features that naturally follow a hierarchical structure:

- **education_level**
(High School < Bachelor < Master < PhD)
- **seniority_level**
(Junior < Senior)

Representing these attributes with ordered numerical values allows the model to capture their relative importance and hierarchical nature, which contributes to more accurate salary estimation.

2. Target Encoding (Major Project Improvement)

One of the most impactful enhancements in the feature engineering pipeline was the implementation of **Target Encoding** for the *job_title* feature.

Instead of assigning arbitrary integers (as in Label Encoding), each job title was encoded using the **mean salary associated with that title** in the training dataset. This approach embeds real economic information directly into the feature space.

Target Encoding improved model behavior in several ways:

- It provided a **more meaningful quantitative representation** of job roles.
- It **reduced overfitting** caused by high-cardinality categorical values.
- It resulted in **more stable and consistent predictions**.

This methodological improvement was examined in detail in the *02_model_training.md* report and was one of the primary reasons for the model's performance increase.

Summary

Overall, the combination of Ordinal Encoding for ordered categories and Target Encoding for high-cardinality job titles created a robust feature representation framework. This allowed the Random Forest model to learn more effectively from the dataset and contributed to the high accuracy achieved during evaluation.

4.2 Model Selection

During the initial phase of the project, several regression models were evaluated, including **Linear Regression**, **Decision Tree Regressor**, and **Random Forest Regressor**. These models were selected for comparison due to their common use in salary prediction problems and their differing assumptions about data structure and feature interactions.

After experimental evaluation, the **Random Forest Regressor** was chosen as the final model for deployment. This decision was based on several factors:

- **Ability to model non-linear relationships:** Salary data does not follow a strictly linear pattern; Random Forest captures complex interactions more effectively.
- **Robustness to outliers:** Tree-based models are less sensitive to extreme values compared to linear models.
- **Compatibility with Target Encoding:** The model performed particularly well when combined with target-encoded categorical features such as job titles.
- **Feature importance interpretability:** Random Forest provides meaningful indicators of which attributes most strongly influence salary predictions.

- **High generalization performance:** Among the tested models, it produced the highest evaluation metrics on the validation set.

The final model configuration used in this project was:

- **n_estimators:** 100
- **random_state:** 42
- **train/test split ratio:** 80% training, 20% testing

This configuration offered a strong balance between predictive accuracy, computational efficiency, and model stability.

4.3 Model Training Workflow

The figure below illustrates the main steps of the model training workflow implemented in the project, as documented in the notebook located at *notebooks/03_model_training.ipynb*.

This workflow includes several key components of the modeling pipeline:

- **Feature Encoding:** Ordinal encoding for education and seniority levels, followed by target encoding for job titles using mean salary values derived from the training set.
- **Train–Test Split:** The dataset is divided into an 80% training set and a 20% testing set to ensure reliable model evaluation.
- **Feature Selection:** Only the engineered and relevant features are selected to form the final training matrix.
- **Preparation for Model Fitting:** Encoded features are applied consistently to both training and testing partitions to avoid data leakage.

This figure highlights the logical structure and modularity of the training process, demonstrating a clear and reproducible workflow prior to fitting the Random Forest model.

```

[13... #Çözümleme
education_map = {
    "unknown": 0,
    "high_school": 1,
    "bachelor": 2,
    "master": 3,
    "phd": 4
}
df['education_level_encoded'] = df['education_level'].map(education_map)

seniority_map = {
    'junior': 0,
    'senior': 1
}
df['seniority_level_encoded'] = df['seniority_level'].map(seniority_map)

[14... # X: Maaşı tahmin etmek için kullanacağımız bilgiler
# y: Hedefimiz (Maaş)
X = df.drop(columns=['salary'])
y = df['salary']

[15... # Veriyi %80 Eğitim, %20 Test diye ikiye bölünmesi
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[16... # Her mesleğin eğitim setindeki ortalama maaşının hesaplanması
job_title_means = y_train.groupby(X_train['job_title']).mean()
global_mean = y_train.mean()

[17... def encode_job_title(title):
    return job_title_means.get(title, global_mean)

[18... X_train['job_title_encoded'] = X_train['job_title'].apply(encode_job_title)
X_test['job_title_encoded'] = X_test['job_title'].apply(encode_job_title)

[19... cols_to_use = [
    'age',
    'years_experience',
    'education_level_encoded',
    'seniority_level_encoded',
    'job_title_encoded'
]

X_train_final = X_train[cols_to_use]
X_test_final = X_test[cols_to_use]

```

Figure 3 - Model training workflow code

5. Results

5.1 Model Performance

The model demonstrated strong predictive capability on the processed dataset. Table 5.1 presents the evaluation results obtained using the test set.

Table 5.1 – Model Performance Metrics

Metric	Score
R ² Score	0.8904
Mean Absolute Error (MAE)	\$11,546

Interpretation:

- The model explains approximately **89% of the variance** in salary values, which indicates a strong overall performance.
- The average prediction error of roughly **\$11.5k** is acceptable given the wide variability across different job roles, education levels, and experience profiles.

5.2 Feature Importance

The Random Forest model also provides insight into which features most strongly influence salary predictions.

Table 5.2 – Feature Importance Values

Feature	Importance
Years of Experience	74.1%
Job Title	18.5%
Age	4.8%
Education Level	1.8%
Seniority Level	0.8%

5.3 System Output (UI Results)

The deployed system provides multiple interactive elements that allow users to obtain personalized salary predictions and compare them with real-world observations. The interface is designed to be simple, intuitive, and informative.

The system includes:

- Estimated salary output
- A confidence interval representing prediction uncertainty
- Closest-matching real employee from the dataset
- A market distribution scatter plot

AI-Based Salary Prediction System

Gazi University - Group 15

Career Details

Job Title: **UX Designer**

Education: **PhD** Seniority: **Junior**

Age: **25** Experience (Years): **16**

CALCULATE & COMPARE

Figure 4 - Career Details Input Form

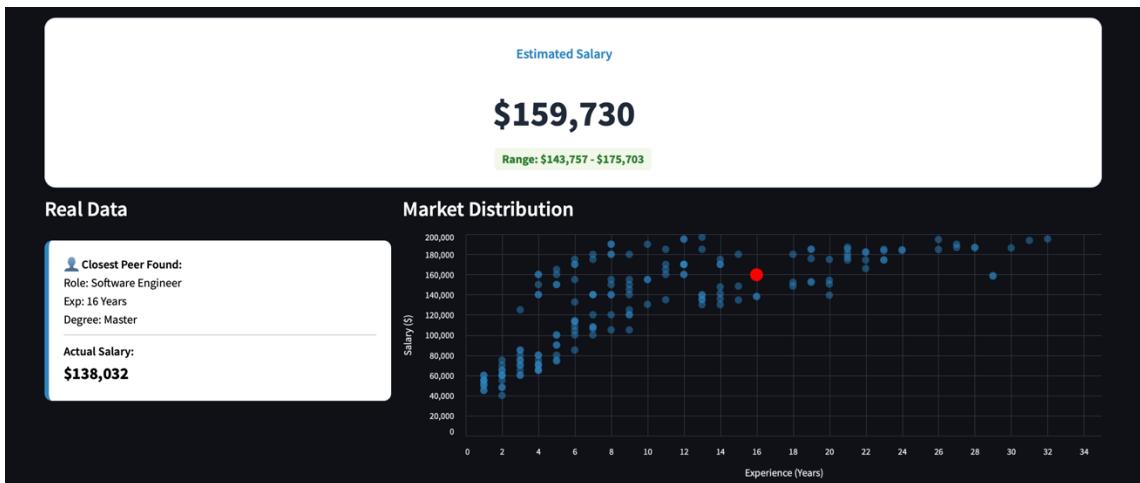


Figure 5 - Salary Prediction Output and Market Distribution

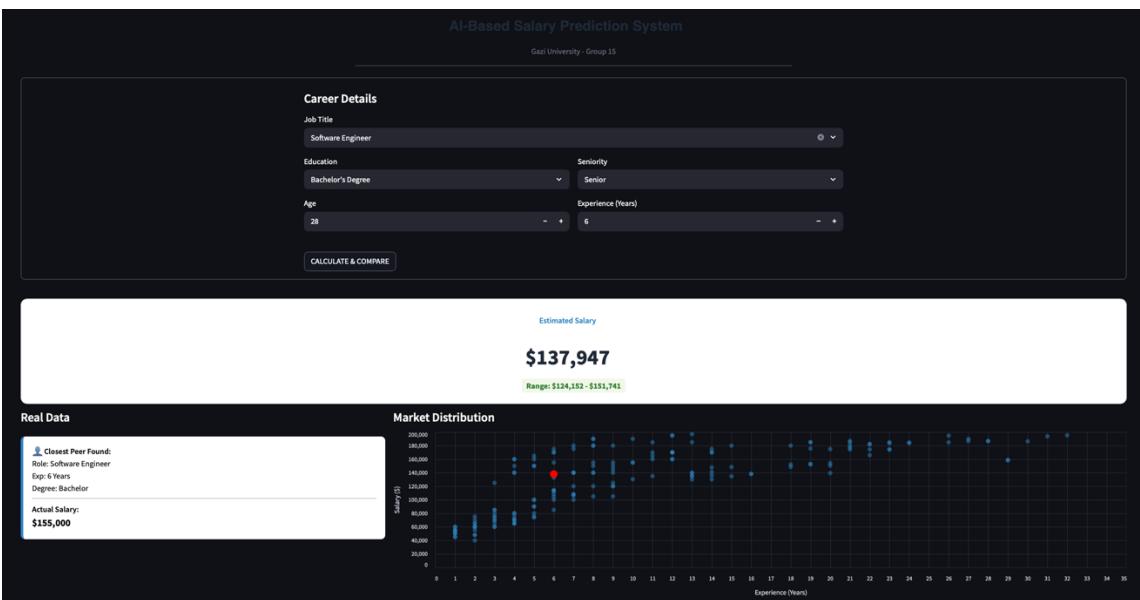


Figure 6 - Salary Prediction Output and Real Data Matching

6. Evaluation

6.1 Strengths

- High model accuracy ($R^2 = 0.89$)
- Predictions are interpretable and grounded in real patterns
- Thorough data preprocessing ensures consistency
- Interactive UI enhances user understanding
- Real-data matching feature increases trust
- Target Encoding significantly improved numerical stability
- Full end-to-end workflow implemented successfully

6.2 Weaknesses & Limitations

- Significant data reduction due to quality issues in raw datasets
- Lack of geographic and currency normalization may reduce accuracy in international predictions
- Job title diversity in open-source datasets may not fully reflect real-world industry variation
- Model performance is inherently limited by the realism and distribution of the available data

6.3 Ethical Considerations

- No personally identifiable information (PII) is included in the dataset
- Salary predictions are estimations, not deterministic outputs
- Age, education, and other demographic factors are treated carefully to avoid unfair bias
- All data was processed and evaluated in compliance with the ethical guidelines of the course

6.4 Future Work

Future improvements may include:

- Integration of geographic cost-of-living adjustments
- Employer dashboard for batch salary analysis
- Support for department-level benchmarking
- Testing more advanced models such as Gradient Boosting or XGBoost

7. Conclusion

This project successfully demonstrates a complete end-to-end data science workflow, including data preprocessing, feature engineering, model development, performance evaluation, and deployment through an interactive user interface.

The final system incorporates:

- A clean and reliable processed dataset
- A robust machine learning model with strong accuracy
- A user-friendly UI for real-time salary predictions
- A real-data matching mechanism that enhances interpretability

With these components, the system is fully ready for presentation and practical demonstration.

References

Project Documentation

- Project Proposal Form: AI-Based Salary Prediction System.* (2025).
- Milestone Report: AI-Based Salary Prediction System.* (2025).
- Project Roadmap.* (2025).
- Data Processing Report.* (2025). Salary Recommendation System Project.
- Model Training Report.* (2025). Salary Recommendation System Project.
- Interface Report.* (2025). Salary Recommendation System Project.

Datasets

- Salary_Data.csv.* (2025). Salary Recommendation System Project.
- Salary by Job Title and Country.csv.* (2025). Salary Recommendation System Project.

Machine Learning & Statistical Methods

- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning* (2nd ed.). Springer.
- Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. Springer.
- Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Data Cleaning & Feature Engineering

- Pyle, D. (1999). *Data preparation for data mining*. Morgan Kaufmann.
- Kelleher, J., & Tierney, B. (2018). *Data science: Statistical and computational methods*. Chapman & Hall.

Model Evaluation & Regression Metrics

- Willmott, C. J., & Matsuura, K. (2005). Advantages of the MAE over the RMSE in assessing average model performance. *Climate Research*, 30, 79–82.

User Interface & Visualization

- McKinney, W. (2018). *Python for data analysis* (2nd ed.). O'Reilly Media.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95.

AI Tools Used During Development

- OpenAI. (2024). *ChatGPT* (Version GPT-5.1) [Large language model]. Assistance used for debugging, documentation refinement, and explanation support. <https://chat.openai.com>
- Google. (2024). *Gemini* [Large language model]. Used for supplemental verification, troubleshooting, and clarification. <https://gemini.google.com>