



ALBUKHARY INTERNATIONAL UNIVERSITY

ALBUKHARY INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING AND INFORMATICS

| COURSE DETAILS | |
|----------------|-------------------------------------|
| SCHOOL | SCHOOL OF COMPUTING AND INFORMATICS |
| COURSE NAME | MACHINE LEARNING |
| COURSE CODE | CCS2213 |
| ASSIGNMENT | INDIVIDUAL ASSIGNMENT 1 |
| SEMESTER | SEM 2 YEAR 2, 2024-2025 |
| NAME | ARIA FIRMANSYAH |
| ID | AIU 22102315 |
| GROUP | A |

| | |
|------------------------------------------|----|
| 1.0 Dataset Background | 3 |
| 1.1 Features and Target | 3 |
| 1.2 Literature Review | 3 |
| 1.3 Class Distribution | 4 |
| 1.4 Balance Evaluation | 4 |
| 2.0 Pre-processing options | 5 |
| 2.1 Handling Unnecessary Columns | 5 |
| 2.2 Salary Value Cleaning | 5 |
| 2.3 Creating the Target Variable | 5 |
| 2.4 Encoding Categorical Variables | 5 |
| 2.5 Feature Scaling | 5 |
| 2.6 Train-Test Splitting | 6 |
| 3.0 Model Evaluation Technique | 7 |
| 3.6 Comparison and Justification | 7 |
| 4.0 Choice of the classifiers | 8 |
| 4.4 Evaluation Metrics | 8 |
| 5.0 Conclusion | 9 |
| 6.0 reference | 10 |

1.0 Dataset Background

The dataset used in this study is titled "**Data Science Fields Salary Categorization**," obtained from Kaggle. The user **whenamancodes** contributed it, which consists of structured data representing job-related information for various positions in the field of data science. This includes job titles, experience levels, salaries (in Indian Rupees), and working conditions such as remote working ratios and company size.

This dataset is suitable for solving classification problems, particularly the categorization of salary levels based on job and company-related features. It contains 607 records and 9 meaningful attributes (after removing the index column), making it manageable and ideal for supervised learning tasks.

1.1 Features and Target

The dataset includes information on job roles and salary details. Key features are the **year** the salary was reported, **job designation**, and **experience level**. It also captures the **employment status**, **annual salary** (in Indian Rupees), and the **locations** of both the employee and the company. Additionally, it records the **company size** (small, medium, or large) and the **remote working ratio**, indicating the percentage of work done remotely.

A new column, **salary_category**, was created to convert the continuous Salary_In_Rupees values into discrete classes for classification purposes:

- **Low:** Salary < ₹5,000,000
- **Medium:** Salary between ₹5,000,000 and ₹10,000,000
- **High:** Salary > ₹10,000,000

This categorization transforms the problem from regression to multi-class classification, in alignment with the assignment objectives.

1.2 Literature Review

Martín et al. (2018) conducted a case study on the Spanish IT job market using Random Forest and AdaBoost classifiers to predict salary ranges. Despite the small dataset and high dimensionality, they achieved strong accuracy with ensemble methods, highlighting the effectiveness of tree-based approaches like Random Forest in salary classification.

Krishna Gopal et al. (2023), in their review of salary prediction in the data science field, evaluated classification algorithms including Decision Tree (J48) and K-Nearest Neighbors (KNN). They found that Decision Tree provided the highest overall accuracy, while KNN also performed reliably, especially when the feature complexity was reduced.

A study by the Grenze Scientific Society (2020) compared KNN, Decision Tree, and Naïve Bayes for salary prediction using the UCI census dataset. KNN achieved the highest accuracy at 87.81%, followed by Decision Tree at 83.10%. These results further

validate the selection of all three models—Decision Tree, KNN, and Random Forest—for salary classification tasks in this project.

1.3 Class Distribution

After converting the salary values into three categories, the class distribution of the target variable (salary category) is as follows:

| Salary Category | Number of Records |
|-----------------|-------------------|
| Low | 153 |
| medium | 233 |
| High | 211 |

The thresholds used to define the salary_category variable were chosen based on the distribution of salary values in the dataset. Exploratory analysis showed that the 25th percentile was approximately ₹5,000,000 and the 75th percentile was around ₹10,000,000.

1.4 Balance Evaluation

Although the dataset is not perfectly balanced, it is considered **moderately balanced**. The Medium and High classes have almost equal representation, while the Low class is somewhat underrepresented but still comprises approximately 25% of the dataset. This distribution is generally acceptable for multi-class classification.

However, due to the slight imbalance, evaluation metrics such as **F1-score**, **precision**, **recall**, and the **confusion matrix** will be more appropriate than relying solely on accuracy. These metrics provide a clearer picture of model performance across all classes, especially when class sizes differ.

2.0 Pre-processing options

Pre-processing is a crucial step in any machine learning pipeline, as it ensures that the data is clean, consistent, and suitable for model training. For this study, the following pre-processing strategies were applied:

2.1 Handling Unnecessary Columns

The original dataset included an unnamed index column that did not carry useful information. This column was removed using:

```
df = df.drop(columns=['Unnamed: 0'])
```

This step ensures that the model is not trained on irrelevant or redundant features.

2.2 Salary Value Cleaning

The Salary_In_Rupees column was originally stored as a string, including comma separators, which prevents numerical operations. It was cleaned and converted into numeric format:

```
df['Salary_In_Rupees'] = df['Salary_In_Rupees'].str.replace(',', '')  
df['Salary_In_Rupees'] = pd.to_numeric(df['Salary_In_Rupees'])
```

This allowed for descriptive statistics and enabled transformation into salary categories.

2.3 Creating the Target Variable

To convert the problem from regression to classification, a new column salary_category was created by binning the Salary_In_Rupees values into three categories based on percentile ranges:

- Low: Salary < ₹5,000,000
- Medium: ₹5,000,000 ≤ Salary ≤ ₹10,000,000
- High: Salary > ₹10,000,000

This transformation supports multi-class classification and balances the dataset reasonably well.

2.4 Encoding Categorical Variables

Several features in the dataset are categorical (e.g., Designation, Experience, Company_Size). These were converted into numerical format using one-hot encoding:

```
df_encoded = pd.get_dummies(df, columns=[ 'Designation', 'Experience',  
'Employment_Status', 'Employee_Location', 'Company_Location', 'Company_Size'  
], drop_first=True)
```

The use of drop_first=True helps avoid the dummy variable trap (multicollinearity), which is important for models sensitive to linear dependencies.

2.5 Feature Scaling

For algorithms that rely on distance measures (e.g., KNN), feature scaling is necessary. The StandardScaler was applied to standardize the numeric features to zero mean and unit variance:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

This ensures fair treatment of all numerical features during model training, especially for KNN.

2.6 Train-Test Splitting

The dataset was split into training and testing sets using **stratified sampling** to preserve the class distribution:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42)
```

This approach ensures that all classes are proportionally represented in both training and test sets, reducing sampling bias and improving generalizability.

3.0 Model Evaluation Technique

In this study, both hold-out validation and 5-fold cross-validation were applied to evaluate the classifiers.

Initially, **hold-out validation** was used by splitting the dataset (607 records) into 70% training and 30% testing sets, with stratification to maintain class balance. This method was chosen for its simplicity and suitability for smaller datasets.

To evaluate model performance, multiple metrics were used due to the class imbalance: **accuracy, precision, recall, F1-score**, and the **confusion matrix**, offering a more nuanced understanding than accuracy alone.

Later, **5-fold cross-validation** was conducted for more stable performance estimates. The results showed that **Random Forest** had the highest average accuracy (0.6097), outperforming both **Decision Tree** (0.5931) and **K-Nearest Neighbors (KNN)** (0.5485). Random Forest's consistent performance highlighted its robustness and generalization ability, while KNN's fluctuating results indicated sensitivity to feature distribution, even after scaling

.

3.6 Comparison and Justification

Hold-out was useful for quick testing and visual analysis, while cross-validation provided more reliable and stable performance metrics. Random Forest consistently outperformed other models in both evaluation methods. Cross-validation reduced the chance of bias from a single data split. Although Decision Tree gave interpretable results, KNN showed lower reliability on this dataset.

4.0 Choice of the classifiers

Three classifiers were selected based on their suitability for the dataset and support from existing studies:

K-Nearest Neighbors (KNN): Selected for its effectiveness on small, scaled datasets like this one. Literature highlights its high accuracy in similar tasks.

Random Forest: Preferred for its robustness, high accuracy, and ability to manage noisy data. It performed best overall and is backed by strong results in related studies.

4.4 Evaluation Metrics

All models were evaluated using:

- Accuracy
- Confusion Matrix
- Precision, Recall, and F1-Score (per class)

These metrics provide a complete picture of model performance, especially in handling slightly imbalanced classes.

5.0 Conclusion

Random Forest emerged as the best-performing model for classifying salary levels in the data science field, showing strong accuracy and generalization. while KNN was less consistent. Overall, tree-based models proved effective for this moderately sized, structured dataset, with proper preprocessing and evaluation ensuring reliable results.

6.0 reference

- Lothe, D. M., Tiwari, P., Patil, N., Patil, S., & Patil, V. (n.d.). *SALARY PREDICTION USING MACHINE LEARNING*. <https://doi.org/10.51319/2456-0774.2021.5.0047>
- Martín, I., Mariello, A., Battiti, R., & Hernández, J. A. (2018). Salary prediction in the IT job market with few high-dimensional samples: A Spanish case study. *International Journal of Computational Intelligence Systems*, 11(1), 1192–1209. <https://doi.org/10.2991/IJCIS.11.1.90>
- Matbouli, Y. T., & Alghamdi, S. M. (2022). Statistical Machine Learning Regression Models for Salary Prediction Featuring Economy Wide Activities and Occupations. *Information 2022*, Vol. 13, Page 495, 13(10), 495. <https://doi.org/10.3390/INFO13100495>
- The Grenze Scientific Society. (2020). Census employee salary prediction using supervised machine learning. *GIJET: Grenze International Journal of Engineering and Technology*, 6(2), 740–745. <https://thegrenze.com/pages/servej.php?id=740>