DATA JOB SALARY PREDICTION

ML-POWERED SALARY INSIGHTS

DEBREBIRBAN UNIVERSITY
COLLEGE OF COMPUTING
DEPARTEMENT OF DATASCIENCE

Course: MACHINE LEARNING

GROUP MEMBER NAME	ID
NATNAEL BEKELE	DBU1501407
HAFIZ HUSEN	DBU1501241
REDIET ESUBALEW	DBU1501704
BEREKET GETAW	DBU1501044
DAWIT ALEMU	DBU1501117
YIFERU MEKONEN	DBU1501562
ELBETEL ABEDI	DBU1501145
GENET MINDA	DBU1501217

Submitted to: inst.Beyenech
Submitted date:04/05/2025

TABLE OF CONTENT

CHAPTER ONE

1. INTRODUCTION

1.1. Background

Machine learning (ML) is a field of artificial intelligence (AI) that enables computers to learn patterns from data and make predictions or decisions without explicit programming. One common ML application is salary prediction, which helps job seekers, employers, and policymakers understand salary trends and influencing factors.

This study focuses on developing an ML model to predict salaries based on various job-related attributes. By leveraging historical salary data, the model aims to provide accurate salary estimations, supporting better career and hiring decisions.

1.1.1. About the Data set

The dataset used in this project was sourced from **Kaggle** and provides comprehensive information on job roles, salaries, and work settings within the data industry. It is designed to support the analysis and prediction of salary trends based on a variety of professional and organizational features.

The dataset used in this study contains 9,355 records of salary-related information. It includes key features such as job title, experience level, employment type, company location, work setting, and salary (both in local currency and USD).

1.2. Statement of the Problem

Salary prediction is a complex problem due to various influencing factors, including job roles, experience levels, company size, and geographic location. Professionals often struggle to estimate fair salaries, while employers need data-driven insights to offer competitive compensation.

Despite the availability of salary data, manually analyzing vast amounts of information is impractical. This study aims to address the following research questions:

1. How do experience level, job title, and location affect salary predictions?
2. Can an ML model provide accurate salary estimations?
3. What factors contribute most to salary variations?

1.3. Objectives

1.3.1. General Objective

The main objective of this study is to develop an ML model that accurately predicts salaries based on job-related features.

1.3.2. Specific Objectives

To achieve the general objective, the study will:

- Analyze the data set to understand key salary determinants.
- Perform data preprocessing, including handling missing values and encoding categorical variables.
- Train and evaluate different ML models for salary prediction.
- Compare model performance using appropriate metrics (e.g., RMSE, MAE).
- Interpret feature importance to identify key salary determinants.

1.4. Significance of the Study

Accurate salary predictions can benefit multiple stakeholders:

Job Seekers: Helps professionals negotiate fair salaries.
Employers: Supports data-driven compensation planning.
Researchers & Economists: Provides insights into salary trends across industries and locations.

This study uses machine learning to predict salaries automatically, helping companies make better decisions about pay and support fair compensation for employees.

1.5. Methodology

1.5.1 Literature Review

A comprehensive review of existing research on salary prediction using machine learning techniques will be conducted to identify established methodologies, best practices, and commonly used features. This review will inform the selection of appropriate algorithms, feature engineering strategies, and evaluation metrics for the current study. Particular attention will be given to studies within the data science and technology sectors to ensure relevance to the dataset at hand.

1.5.2 Tools & Techniques

Programming Language: Python (with libraries like Pandas, Scikit-learn, TensorFlow).

ML Models: Linear Regression, Decision Trees, Random Forest, XGBoost AND Other .

Evaluation Metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE).

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

Salary prediction is an important application of machine learning (ML) in human resources and workforce analytics. Organizations, job seekers,recruiters, and policy makers increasingly rely on data-driven methods to assess market-aligned compensation. As digital recruitment platforms and online job portals collect extensive employment-related data, ML has become a valuable tool for uncovering patterns and building predictive salary models based on job roles, qualifications, experience, and geographical trends.

Traditional compensation analysis was often based on static benchmarking reports or manually aggregated surveys. However, these methods lack adaptability and fail to capture real-time market dynamics. In contrast, ML offers dynamic, scalable, and automated prediction capabilities, enabling the estimation of salaries with greater precision. This is especially relevant in fields like data science, software engineering, and remote work, where salaries can vary dramatically depending on specific features such as skillset, company location, and employment type (e.g., contract vs. full-time).

Several ML algorithms have been applied to the salary prediction problem, each offering distinct advantages depending on the dataset characteristics and feature complexity. The most widely used approaches include:

1. Linear Regression

Linear regression is one of the simplest and most interpretable models used for regression tasks, including salary prediction. It assumes a linear relationship between independent variables (e.g., job title, experience level) and the dependent variable (salary). While easy to implement and explain, its performance may suffer when handling complex interactions, non-linearity, or high-dimensional categorical data.

2. Decision Trees

Decision trees split the dataset based on the most informative features to create a model that resembles a flowchart. For salary prediction, decision trees offer a transparent way to explore how certain features (like years of experience or company location) influence outcomes. They are prone to overfitting, but when tuned properly or used in ensembles, they perform well on mixed-type data.

3. Random Forest

Random Forest is an ensemble technique that combines multiple decision trees to improve accuracy and reduce overfitting. It works well for datasets with a mix of categorical and continuous variables. In salary prediction tasks, Random Forest can identify key determinants of salary, provide robust predictions, and measure feature importance.

4. Gradient Boosting Machines (GBM, including XGBoost and LightGBM)

Boosting algorithms build models sequentially, correcting the errors of previous models. XGBoost and LightGBM are popular implementations that are highly efficient and accurate for tabular data. These models can capture complex feature interactions and are often considered top choices for salary prediction tasks due to their superior performance in competitive ML environments.

5. Support Vector Regression (SVR)

SVR attempts to fit a hyperplane that captures most data points within a margin of tolerance. It is useful when the data has a high number of features or when there is non-linear behavior. Kernel functions (like RBF) help SVR manage non-linearity, though it may not scale well to very large datasets.

7. K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm. It predicts the salary of a given job instance by averaging the salaries of its closest neighbors. Though simple, KNN is computationally expensive with large datasets and often less accurate than ensemble methods.

2.2 Types of Machine Learning for Salary Prediction

Machine Learning can be broadly categorized into three types:

1. Supervised Learning

In supervised learning, the model learns from labeled data. This is the most common type used in salary prediction, where the input features (job title, experience level, etc.) are used to predict a known output (salary). Algorithms like Linear Regression, Decision Trees, and XGBoost fall under this category.

2. Unsupervised Learning

In unsupervised learning, the data has no labeled outcomes. While not commonly used for direct salary prediction, unsupervised techniques such as clustering (e.g., K-means) can group similar job roles or salary bands for exploratory purposes.

3. Reinforcement Learning

This is used when an agent learns through interaction with an environment by receiving feedback. Though powerful in fields like robotics and gaming, it is rarely applied in salary prediction tasks.

2.3 Key Components of an ML System

A typical machine learning system involves the following core components:

- **Data:** The foundation of any ML model. For salary prediction, this includes job roles, salaries, locations, etc.
- **Features**: Variables or attributes extracted from the raw data that are relevant to the task.
- **Model**: The algorithm that maps input features to output predictions.
- **Loss Function**: A measure of how far the predicted values are from actual values (e.g., Mean Squared Error).
- **Evaluation Metrics**: Used to assess model performance (e.g.,RMSE, MAE,R^2 Score).

2.4 Steps in a Machine Learning Project

ML-based salary prediction involves several steps:

1. **Problem Definition**
   Define the business goal, e.g., "predict annual salary in usd based on job features."
2. **Data Collection**
   Gather historical salary datasets (e.g., from Kaggle).
3. **Data Preprocessing**

   - Handle missing values
   - Remove duplicate rows
   - Encode categorical variables
   - Normalize or scale features
   - Handle outliers

4. **Exploratory Data Analysis (EDA)**
   Use statistics and visualizations to understand feature distributions and correlations.
5. **Feature Selection**
   Feature selection removed redundant and target-related features (e.g., salary currency) to reduce multicollinearity and prevent data leakage, improving model generalization.
6. **Model Selection and Training**
   Choose appropriate algorithms and train them using training data.
7. **Model Evaluation**
   Assess the model using test data and metrics like RMSE and R^2.
8. **Interpretation & Deployment**
   Interpret results, identify important features, and integrate the model into real systems.

2.5 Methods Commonly Used in Salary Prediction

Some of the most popular methods and tools in salary prediction

Method	**Description**
Linear Regression	Assumes a linear relationship; simple and interpretable.
Decision Trees	Splits data into nodes; good for feature importance.
Random Forest	Ensemble of trees; robust and handles overfitting.
XGBoost / LightGBM	Advanced gradient boosting; high accuracy in tabular data.
SVR (Support Vector)	Fits data within margins; good for small or high-dimensional datasets.
KNN Regression	Predicts by averaging nearby salaries; intuitive but computationally heavy.

CHAPTER THREE

DATA DESCRIPTION AND PREPROCESSING

3.1 Dataset Overview

In any machine learning project, the foundation of success lies in the quality and relevance of the data. This project uses a publicly available dataset sourced from Kaggle, which contains **9,355 records** pertaining to employee salaries in the technology and data science sectors. The dataset represents a global workforce, making it suitable for drawing generalized insights into compensation structures across job titles, experience levels, and geographical regions.

Each record in the dataset includes the following key features:

> **work_year**: The year in which the data was recorded. This field indicates the temporal context of the data, important for understanding salary trends over time.
>
> **job_title**: The specific title of the job role, like 'Data Scientist', 'Data Engineer', or 'Data Analyst'. This column is crucial for understanding the salary distribution across various specialized roles within the data field.
>
> **job_category**: A classification of the job role into broader categories for easier analysis. This might include areas like 'Data Analysis', 'Machine Learning', 'Data Engineering', etc.
>
> **salary_currency**: The currency in which the salary is paid, such as USD, EUR, etc. This is important for currency conversion and understanding the actual value of the salary in a global context.
>
> **salary**: The annual gross salary of the role in the local currency. This raw salary figure is key for direct regional salary comparisons.
>
> **salary_in_usd**: The annual gross salary converted to United States Dollars (USD). This uniform currency conversion aids in global salary comparisons and analyses.

employee_residence: The country of residence of the employee. This data point can be used to explore geographical salary differences and cost-of-living variations.

experience_level: Classifies the professional experience level of the employee. Common categories might include 'Entry-level', 'Mid-level', 'Senior', and 'Executive', providing insight into how experience influences salary in data-related roles.

employment_type: Specifies the type of employment, such as 'Full-time', 'Part-time', 'Contract', etc. This helps in analyzing how different employment arrangements affect salary structures.

work_setting: The work setting or environment, like 'Remote', 'In-person', or 'Hybrid'. This column reflects the impact of work settings on salary levels in the data industry.

company_location: The country where the company is located. It helps in analyzing how the location of the company affects salary structures.

company_size: The size of the employer company, often categorized into small (S), medium (M), and large (L) sizes. This allows for analysis of how company size influences salary.

This structure enables both exploratory analysis and model training for predicting salary using a wide variety of real-world features.

3.2 Data Cleaning and Preparation

Data preprocessing is a critical phase in machine learning pipelines. It ensures that data fed into the model is accurate, consistent, and in a format suitable for modeling. The following steps were taken to prepare the dataset:

3.2.1 Handling Duplicates

Duplicate rows were handled both before and after addressing missing values to ensure data integrity at each stage of preprocessing.

- **Before Handling Nulls**: Initial duplicate removal was performed to eliminate exact duplicate records, ensuring that statistics used for imputing missing values (e.g., mean, median, mode) were not biased by redundant data.
- **After Handling Nulls**: A second duplicate check was conducted after missing values were imputed or removed. This step ensured that the imputation process did not result in new duplicate rows, especially when multiple records had missing values filled with the same replacement values.

3.2.2 Handling Missing Values

Missing data was handled with a combination of **conditional imputation** and **column-wise removal**, based on the completeness and relevance of each attribute.

- **Removing Rows with Null Job Titles**
  Rows with missing job_title values were dropped, as these typically had multiple missing fields and lacked essential information required for analysis. Since the job title is a critical feature in salary prediction, retaining such incomplete rows would compromise data quality.
- **Imputing** salary_in_usd **When Currency Is USD**
  For cases where salary_in_usd was missing but the salary_currency was "USD" and a value was present in the salary column, the missing salary_in_usd was replaced with the corresponding salary value. This imputation was valid because no currency conversion was needed in such instances.
- **Dropping Redundant or Sparse Columns**
  The salary column was dropped after imputation due to a high percentage of missing values (approximately 41%), which limited its usefulness for modeling. Additionally, salary_currency was removed, as it became redundant after converting all salaries to USD and eliminating the original salary field.

3.3 Feature Engineering

Feature engineering refers to the process of creating new input variables (features) from existing data that improve the predictive power of models. The following techniques were used in this study:

3.3.1 Skewness and Log Transformation

- The salary_in_usd variable exhibited a skewness of **0.29**, which falls in the range of **fairly symmetrical**. Although not highly skewed, applying a **log transformation** can still be beneficial for stabilizing variance, improving normality, and enhancing model performance—especially for algorithms sensitive to data distribution.
- Log transformation helps reduce the effect of potential outliers and compresses the scale of high salary values, making the data more suitable for regression models

3.3.2 Outlier Detection and Winsorization

- Outliers in salary_in_usd were treated using **winsorization** within each group of experience_level, employment_type, work_setting, and company_size. Salaries below the 5th percentile and above the 95th percentile in each group were capped at those thresholds.
- **Winsorization** was chosen because it reduces the impact of extreme values without removing data points, preserving dataset size and improving model stability and performance.

3.3.3 Target Scaling

- The target variable (*salary_in_usd*) was scaled using StandardScaler to normalize its distribution (mean = 0, standard deviation = 1). This helps improve model performance and convergence. The training data was scaled with fit_transform, and the same parameters were applied to the test set using transform. Reshaping was done to match the expected input format for scaling.

3.3.4 Feature Selection

The *employee_residence* column was dropped due to 97.8% overlap with *company_location*. Removing this redundant feature simplifies the model and prevents multicollinearity without losing important information.

3.3.5 Encoding Categorical Variables

- **Frequency Encoding (High Cardinality)**
  - Applied to job_title and company_location, which have many unique values.
  - Replaces categories with their relative frequency in the training data.
  - **Reason**: Avoids creating too many one-hot columns, keeps model efficient, and captures useful information about how common a category is.

- **Ordinal Encoding (Ordered Categories)**
  - experience_level and company_size are mapped to integers based on logical order (e.g., Entry < Mid < Senior < Executive).
  - **Reason**: These features have a natural ranking, and mapping preserves that order, helping models interpret magnitude.

- **One-Hot Encoding (Unordered Categories)**

  - Applied to job_category, employment_type, and work_setting.
  - drop_first=True avoids the dummy variable trap (multicollinearity).
  - **Reason**: These variables are nominal (no natural order), so binary columns help the model learn differences without implying rank.

CHAPTER FOUR

MODEL DEVELOPMENT AND EVALUATION

4.1 Introduction

Once the dataset has been properly cleaned, prepared, and enriched through feature engineering, the next critical phase in a machine learning pipeline is **model building**. In this chapter, we describe how different regression models were selected, trained, and evaluated to predict employee salaries. The goal is to identify the model that achieves the best balance between predictive accuracy, interpretability, and computational efficiency.

We also discuss hyperparameter tuning, model validation strategies, and performance measurement using standardized evaluation metrics.

4.2 Model Selection Strategy

The salary prediction problem is a **regression task**, where the target variable (**salary_in_usd**) is continuous. As such, we selected a range of supervised regression algorithms that can capture both linear and non-linear relationships:

1. Linear Regression

A simple and interpretable model that assumes a linear relationship between input features and the salary. It serves as a baseline model for comparison.

2. Decision Tree Regressor

A non-linear model that splits the dataset into decision rules based on feature thresholds. It helps visualize which features play major roles in predicting salary.

3. Random Forest Regressor

An ensemble of decision trees, trained using the bagging method. Random Forest improves generalization by averaging the predictions of multiple trees.

4. XGBoost Regressor

A state-of-the-art gradient boosting algorithm known for high accuracy and efficiency. XGBoost builds models sequentially and corrects the errors of previous iterations, making it very effective for structured/tabular data.

5. Support Vector Regression (SVR)

Useful for high-dimensional feature spaces, SVR attempts to fit the data within a specified margin of error. It is sensitive to feature scaling.

6. k-Nearest Neighbors (k-NN) Regressor

A non-parametric model that predicts salary in usd based on the average salaries of the closest job profiles in the feature space. Effective but computationally expensive on large datasets.

4.3 Data Splitting and Validation

To train and evaluate models, the dataset was split as follows:

Training Set (80%): Used to fit the machine learning models.

Testing Set (20%): Used to evaluate the performance of the trained models on unseen data.

4.4 Model Training and Hyperparameter Tuning

Each selected algorithm was trained using the preprocessed dataset, and hyperparameters were fine-tuned using **Grid Search** and **Randomized Search** approaches:

➢ Linear Regression:

- fit_intercept: Whether to calculate the intercept for the model.
- normalize: Whether to normalize input features (deprecated in newer versions of scikit-learn).
- n_jobs: Number of CPU cores used during model training (for parallelism in some variants like Ridge, Lasso).

- **Decision Tree:**

  - max_depth: Maximum depth of the tree; controls model complexity.
  - min_samples_split: Minimum number of samples required to split an internal node.
  - min_samples_leaf: Minimum number of samples required to be at a leaf node.
  - criterion: Function to measure the quality of a split (e.g., "squared_error" for regression).
  - max_features: Number of features to consider when looking for the best split.

- **Random Forest**:

  - **n_estimators:** Number of trees in the forest.
  - **max_depth**: Controls the depth of each tree.
  - **min_samples_split**: Minimum number of samples required to split a node.

- **XGBoost**:

  - **learning_rate**: Controls the contribution of each tree.
  - **n_estimators**: Total number of boosting rounds.
  - **max_depth**: Maximum depth of a tree.
  - **subsample**: Ratio of rows sampled per iteration.

- **SVR**:

  - **kernel**: Type of kernel function (linear, RBF, etc.).
  - **C**: Regularization parameter.
  - **epsilon**: Tolerance for error margin.

- **KNN**:

  - **n_neighbors:** Number of nearest points used for prediction.
  - **weights**: Uniform or distance-based weighting.

4.5 Evaluation Metrics

To evaluate and compare model performance, we used the following metrics:

1. Mean Absolute Error (MAE)

Represents the average absolute difference between predicted and actual values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

2. Root Mean Squared Error (RMSE)

Penalizes larger errors more than MAE. Commonly used in regression problems.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

3. R-squared (R^2 Score)

Indicates the proportion of variance in the target variable explained by the model. Ranges from 0 to 1, where 1 is perfect prediction.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

4.6 Model Comparison and Evaluation Metrics Overview

After training and validating multiple regression models, we compared their performance using two standard evaluation metrics:

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values. Lower values indicate better performance.
- **R^2 Score (Coefficient of Determination)**: Reflects how much of the variance in the target variable is explained by the model. Values closer to 1 indicate higher predictive accuracy.

The results of the evaluation are summarized in the table below:

Model	**MSE (Before)**	**R^2 (Before)**	**MSE (After)**	**R^2 (After)**	**Improvement Summary**
Linear Regression	0.5949	0.3825	0.5943	0.3831	Minimal change due to limited tunable parameters; linear model has low complexity.
Decision Tree	0.6243	0.3520	0.5807	0.3972	Tuning max_depth and min_samples_leaf improved generalization.
XGBoost	0.5448	0.4300	**0.5400**	**0.4395**	Best overall performance after extensive hyperparameter tuning.
Random Forest	0.5734	0.4048	0.5480	0.4311	Ensemble power and depth tuning enhanced performance.
SVR	0.5895	0.3881	0.5833	0.3900	Slight gain with kernel and parameter optimization.
KNN	0.6947	0.2789	0.6439	0.3317	Improvement noted, but model still underperforms in high-dimensional space.

Model Comparison and Observations

➢ Linear Regression

- **R^2 Score:** ~0.38 **MSE:** ~0.594
- **Observation:**
  Linear Regression served as a baseline model but performed the worst among all. Its low R^2 indicates it struggled to capture non-linear relationships or complex feature interactions. While simple and interpretable, it lacks the flexibility needed for this dataset.

➢ Decision Tree

- **R^2 Score:** ~0.40 (after tuning) **MSE:** ~0.581
- **Observation:**
  Decision Tree improved over linear regression by modeling non-linear patterns, but it still exhibited moderate performance. Its tendency to overfit was mitigated through hyperparameter tuning, yet it did not generalize as well as ensemble methods.

➢ Random Forest

- **R^2 Score:** ~0.43 (after tuning) **MSE:** ~0.548
- **Observation:**
  Random Forest showed strong performance, benefiting from ensembling and feature randomness. It modeled complex patterns and avoided overfitting, making it one of the top-performing models. Its balance between accuracy and robustness stands out.

➢ XGBoost

- **R^2 Score:** ~0.44 (after tuning) **MSE:** ~0.540
- **Observation:**
  XGBoost outperformed most models, with the lowest MSE and highest R^2. It handles feature interactions and overfitting efficiently via regularization and boosting. It is especially well-suited for structured/tabular data like this.

- ### Support Vector Regression (SVR)

  - **R^2 Score:** ~0.39 (after tuning) **MSE:** ~0.583
  - **Observation:**
    SVR provided reasonable performance and showed its strength in modeling non-linear relationships using the RBF kernel. However, it was computationally heavier and slightly underperformed compared to tree-based ensembles.

- ### K-Nearest Neighbors (KNN)

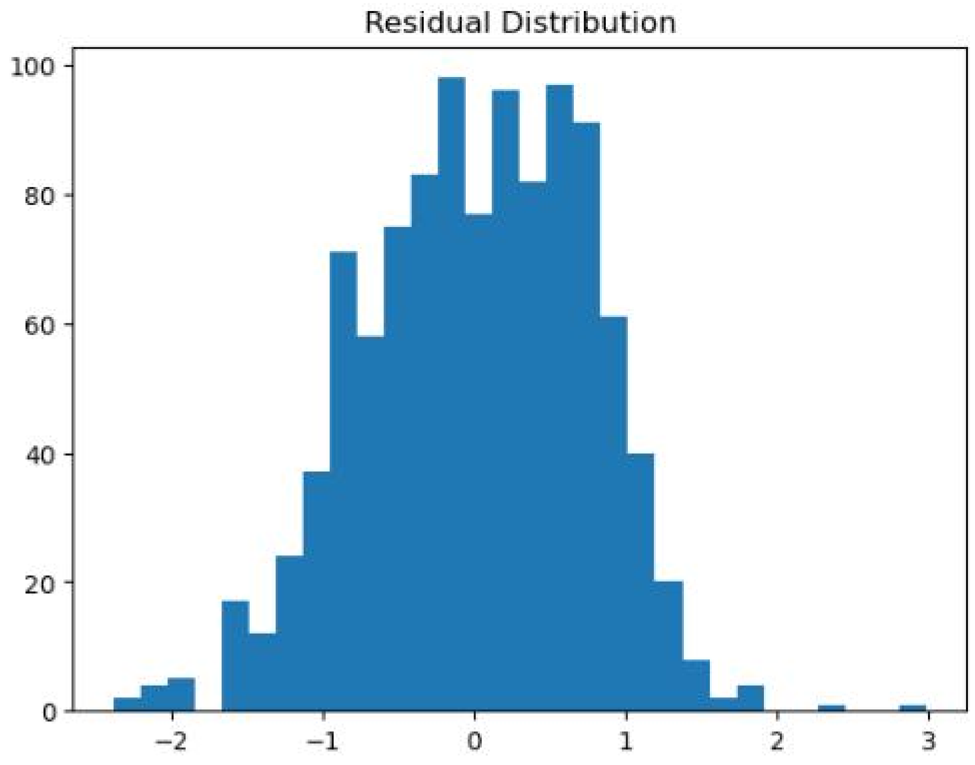
  - **R^2 Score:** ~0.33 (after tuning) **MSE:** ~0.644
  - **Observation:**
    KNN had the lowest performance among non-linear models. Its prediction quality was limited, possibly due to the high-dimensional nature of the data, which dilutes the meaning of distance (curse of dimensionality).

XGBoost and **Random Forest** emerged as the best models overall, offering a strong balance of low error and high generalization. Their ability to capture complex patterns without overfitting makes them ideal for salary prediction on structured data. Linear models, while useful for interpretability, were insufficient in capturing data complexity.

Best Performing Model: XGBoost

- Achieved the **lowest MSE** and **highest R^2 score**.
- Effectively captures complex patterns through boosting and robust hyperparameter tuning.
- Recommended for final deployment based on accuracy and consistency.

4.7 Residual Distribution Analysis



The histogram presented above illustrates the distribution of residuals obtained from the trained regression model. Residuals represent the difference between the actual and predicted values, formally defined as:

Residual=Actual Value − Predicted Value

Analyzing the residuals is a crucial step in evaluating model performance and verifying underlying statistical assumptions.

Interpretation:

➢ **Symmetric, Approximately Normal Distribution:**

- The residuals exhibit a bell-shaped, symmetric distribution centered around zero.
- This suggests that the model does not introduce significant bias and that prediction errors are randomly distributed.

➢ Concentration Around Zero:

- A large proportion of residuals fall within the range of approximately -1 to +1.
- This indicates that the model is generally accurate, with the majority of predictions closely aligned with the actual values.

➢ Minimal Outliers:

- The distribution shows very few extreme residuals (beyond ±2), implying the model produces relatively few large errors.
- The presence of thin tails further supports the reliability of the model across most data points.

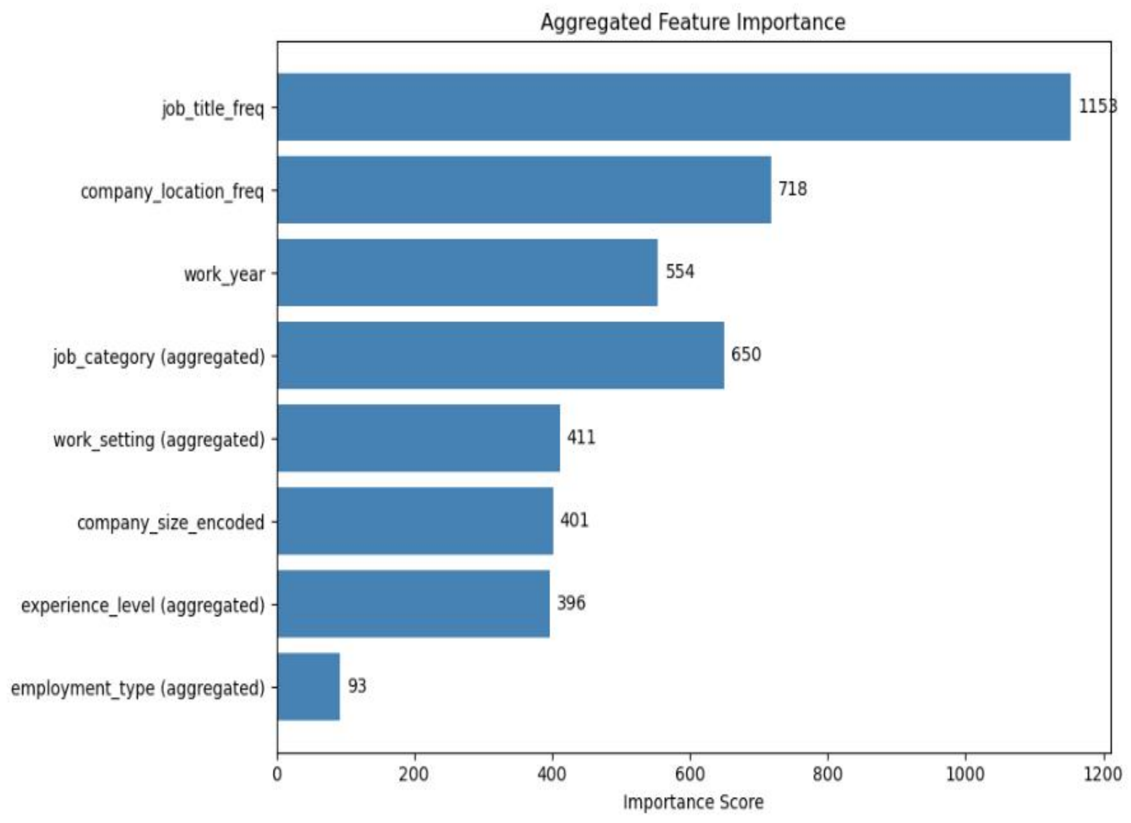
➢ Absence of Skewness:

- The distribution is not significantly skewed to the left or right, indicating that the model does not systematically overpredict or underpredict.
- There is no evidence of heteroscedasticity or other violations of standard regression assumptions based on the residual pattern.

In conclusion The residual distribution provides **strong evidence** that the model is performing **well**. The near-normal shape centered at zero indicates that the prediction errors are random and not systematically biased, supporting the validity of the model’s assumptions. The absence of significant skew or heavy tails also suggests that no major data anomalies are distorting the model's predictions.

4.7 Aggregated Feature Importance Analysis

The following chart presents the aggregated feature importance scores derived from the trained XGBoost regression model. One-hot encoded categorical variables were grouped into their original categories to enhance interpretability. The importance scores reflect each feature's relative contribution to the model's predictive performance.



Key Findings:

1. job_title_freq **(Importance Score: 1153)**

   ➢ This feature has the highest impact on the model's output. It indicates that the frequency or specific type of job title plays a critical role in predicting the target variable. This may reflect market demand, seniority levels, or role-specific value.

2. company_location_freq **(Importance Score: 718)**

   - The geographic location of the company is also a significant predictor, suggesting that regional differences—such as cost of living, labor market conditions, or regional specialization—affect the outcome variable.

3. job_category **(Aggregated Importance Score: 650)**

   - Aggregated from multiple one-hot encoded job category features, this variable demonstrates that the field or specialization of a job (e.g., Data Science, Machine Learning, Engineering) is a major determinant of the target variable.

4. work_year **(Importance Score: 554)**

   - This temporal feature captures variation across different years, reflecting the impact of evolving market trends, technological developments, or broader economic factors (e.g., post-pandemic changes).

5. work_setting **(Aggregated Importance Score: 411)**

   - The mode of work (e.g., remote vs. in-person) contributes meaningfully to the prediction, suggesting that the nature of the working environment influences the outcome, potentially due to differences in compensation structures or flexibility.

6. company_size_encoded **(Importance Score: 401)**

   - Company size, encoded as an ordinal feature, has a moderate impact, indicating that the scale of an organization may affect job outcomes, though not as strongly as the aforementioned factors.

7. experience_level **(Aggregated Importance Score: 396)**

   - This variable reflects the professional level (e.g., entry, mid, senior), showing that experience has a noticeable but secondary influence on the prediction.

8. employment_type **(Aggregated Importance Score: 93)**

   - Representing employment arrangements (e.g., full-time, freelance, part-time), this feature shows the least importance, indicating that employment type has minimal effect on the model’s predictions in this context.

Generally: The aggregated feature importance analysis shows that job-related features—like job title and category—along with company attributes, especially location, and the timing of the data, are the most influential factors in the model. In contrast, employment format and experience level are also important but have a smaller impact on the model's predictions. These findings can help guide feature selection, improve model interpretation, and inform future data collection strategies.

4.8 Limitations and Challenges

The XGBoost model achieved an R^2 score of approximately 0.44, indicating moderate predictive power and explaining less than half of the variance in the target variable. Despite extensive hyperparameter tuning, performance improvements were minimal, suggesting the model is near its capacity given the current data.

Limitations include a potentially incomplete feature set, lacking key variables such as salary, qualifications, and company size. Some features show limited predictive value, and data quality issues—such as inconsistent categories and imbalanced classes—further restrict model accuracy. Additionally, the model does not incorporate external or temporal factors like economic trends or seasonality, which could influence outcomes.

Overall, while the model is stable, its predictive ability is constrained by data and feature limitations. Addressing these issues is crucial for enhancing future model performance and generalizability.

4.9 Recommendations for Future Work

To address the limitations identified and improve model performance, the following recommendations are proposed:

- **Enrich Feature Set:**
  Incorporate additional relevant variables such as salary range, required skills, education level, company size, and job benefits to enhance explanatory power.
- **Expand Dataset Volume and Diversity:**
  Increase data quantity and diversity by collecting more job listings over extended periods and across varied platforms, emphasizing underrepresented categories and geographic regions.
- **Incorporate Contextual and Temporal Variables:**
  Include posting dates, economic indicators, and industry trends to capture temporal dynamics and external market factors.

- **Explore Advanced Modeling Techniques:**
  Consider alternative approaches such as neural networks, ensemble methods, or time series models to better capture complex feature interactions.

Conclusion:
Implementing these recommendations will enhance model accuracy, robustness, and real-world applicability by leveraging richer data, improved features, and advanced analytical methods.

4.10 References

- https://www.youtube.com/watch?v=Y2B7F9IMbqY&t=367s
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7
- Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774.
- Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer.