

**16-Piyush Nimbalkar**

**18-Gauri Chorge**

**9-Sanika Bharne**

Group 4: Regression: Developer’s Salary Prediction.

# Executive Summary

This project aims to build a highly accurate machine learning model to predict developer salaries based on historical data from the Stack Overflow Developer Survey. In a rapidly evolving tech industry, understanding compensation trends is crucial for both job seekers and employers. Developers often face uncertainty regarding fair market compensation, while companies struggle to benchmark salaries effectively. This project addresses these challenges by using regression-based models to predict salaries using features such as country, education level, years of professional coding experience, developer role, and employment status.

The target users of this system include recruiters, HR managers, developers, and career guidance platforms who require reliable salary predictions to make informed decisions. The project follows a structured pipeline that involves data cleaning, preprocessing, feature engineering, exploratory data analysis (EDA), and model training. Multiple regression algorithms were evaluated, including Linear Regression, Random Forest, and XGBoost Regressor. Among these, the XGBoost model demonstrated the best performance, achieving a high R² score and low RMSE, which indicates strong predictive capability.

From a financial standpoint, the project is cost-efficient, relying entirely on open-source libraries and publicly available data. It can be deployed as an API or integrated into platforms like developer job boards and HR analytics dashboards. Overall, the system provides a transparent, interpretable, and scalable solution for salary prediction. The report outlines the methodology in detail, covering data exploration, model evaluation, implementation, costing, and potential applications, while also suggesting improvements and enhancements for future development.

# Table of Contents

[Executive Summary 1](#_Toc49359911)

[Table of Contents 2](#_Toc49359912)

[List of Figures & Tables 3](#_Toc49359913)

[Glossary of Terms 4](#_Toc49359914)

[Introduction 5](#_Toc49359915)

[Literature Survey 6](#_Toc49359916)

Improvement suggested 7

[Design 8](#_Toc49359917)

[Implementation 9](#_Toc49359918)

[Testing & QA 10](#_Toc49359919)

[Project Costing 1](#_Toc49359920)1

[Applications & Future Enhancements 1](#_Toc49359921)2

[References 1](#_Toc49359922)3

[Appendix 1](#_Toc49359923)4

# List of Figures & Tables

|  |  |
| --- | --- |
| **Figure/Table No.** | **Title** |
| **Figure 1** | Distribution of Developer Salaries |
| **Figure 2** | Boxplot: Salary by Country |
| **Figure 3** | Heatmap of Feature Correlation |
| **Figure 4** | Scatter Plot: Years of Coding Experience vs Salary |
| **Figure 5** | Residual Plot of Predictions |
| **Figure 6** | Model Comparison: R² Scores Across Algorithms |
| **Table 1** | Raw Dataset Features Description |
| **Table 2** | Final Features Selected After Encoding & Cleaning |
| **Table 3** | Model Performance Metrics – XGBoost (R² Score, RMSE) |
| **Table 4** | Encoded Feature Columns (e.g., Country → Numeric, Education → Ordinal) |
| **Table 5** | Project Cost Breakdown |

# Glossary of Terms

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | **Definition** | | **Regression** | A machine learning technique used to predict continuous numerical values. | | **R² Score** | The coefficient of determination; measures how well the model fits the data. | | **RMSE** | Root Mean Squared Error; measures the average magnitude of prediction error. | | **EDA** | Exploratory Data Analysis; involves visualizing and summarizing dataset trends. | | **Feature Encoding** | Converting categorical variables into numerical format for ML algorithms. | | **Outliers** | Data points significantly different from others; may distort model accuracy. | | **XGBoost** | An advanced, efficient boosting algorithm used for regression and classification. | | **Normalization** | Scaling numerical data to a standard range, often 0 to 1. | | **Train-Test Split** | Dividing data into training and testing sets to evaluate model performance. | | **Overfitting** | When a model learns noise in training data and performs poorly on new data. | | **Feature Engineering** | Creating or transforming variables to improve model performance. | | **Pickle/Joblib** | Python libraries used to save and load machine learning models. | | **Label Encoding** | Assigning numerical values to categorical labels (e.g., Male = 0, Female = 1). | |

# 

# Introduction

The Developer Salary Prediction project aims to build a robust machine learning regression model that can accurately estimate a software developer’s salary based on various professional and demographic attributes. With the rise of remote work, global job markets, and rapidly evolving tech roles, predicting developer compensation has become increasingly complex and data-driven. This system addresses the challenge by analyzing real-world survey data collected from developers worldwide.

The core objective of this project is to streamline salary estimation using a data-driven, scalable, and interpretable machine learning pipeline. Key features considered for prediction include the developer's country, education level, years of experience, employment type, and technologies used. The project incorporates essential steps like data cleaning, feature encoding, exploratory data analysis (EDA), model training, and performance evaluation.

The system architecture is modular and includes the following components: a **Data Loader** for reading and structuring the dataset; a **Preprocessing Module** to clean missing values and encode categorical features; an **EDA Module** for visual and statistical analysis; a **Model Builder** that implements multiple regression algorithms like Linear Regression, Random Forest, and XGBoost; and an **Evaluation Module** to assess models using metrics such as R² Score and RMSE. The pipeline concludes with a **Visualization Module** to present insights and a **Model Exporter** to save the final model for deployment.

This project demonstrates how machine learning can support decision-making in HR, hiring platforms, and freelance marketplaces by providing accurate and explainable salary predictions tailored to global developers' profiles.

# Literature Survey

Several platforms and studies have attempted to address the challenge of predicting developer salaries, each employing different methodologies and offering varying levels of accuracy and scalability. Understanding existing systems and related work helps position this project within the broader landscape of salary prediction tools and provides guidance on what strategies work best.

One notable reference point is the **Stack Overflow Developer Survey**, which gathers extensive self-reported data from developers worldwide, including salary, education, experience, and tech stack. While this dataset provides a rich foundation for analysis, its limitation lies in inconsistencies and missing values that require substantial preprocessing. Nevertheless, it remains a trusted source for salary insights and trend analysis across regions and job roles.

**Glassdoor** and **Levels.fyi** are two platforms that provide crowdsourced salary information. While these offer real-time insights and compensation breakdowns by company and location, they are largely rule-based, lack ML-based predictions, and sometimes suffer from data sparsity in certain regions or roles. Their closed and proprietary nature also restricts algorithmic customization.

From an academic perspective, past research studies have utilized **linear regression** models due to their simplicity and interpretability. However, they often fall short when handling nonlinear relationships and multicollinearity between features. More recent work has explored **Random Forest** and **Gradient Boosting** models like **XGBoost**, which offer better performance on structured data, robustness to missing values, and the ability to capture complex interactions among variables.

The following table summarizes key systems and their advantages and limitations:

|  |  |  |
| --- | --- | --- |
| **System/Reference** | **Pros** | **Cons** |
| Stack Overflow Survey | Rich, global dataset | Noisy and incomplete data |
| Glassdoor | Real-time data; company-specific insights | Lacks predictive capabilities, limited data access |
| Levels.fyi | Detailed salary breakdown | Not ML-based; focus on big tech only |
| Linear Regression | Interpretable, fast | Poor performance with non-linear patterns |
| XGBoost | High accuracy, handles missing values | Computationally expensive, needs parameter tuning |

This project leverages the strengths of modern machine learning models like XGBoost while overcoming data issues through preprocessing, encoding, and normalization. It offers an open-source, flexible alternative to existing platforms with the added advantage of predictive modeling.

# Improvement suggested

While the current Developer Salary Prediction model demonstrates solid performance using algorithms like XGBoost and Random Forest, several areas for enhancement remain. Firstly, the dataset can be improved by enriching it with additional external features such as company size, industry sector, or location-based cost of living indices to provide more context for salary variation. Secondly, outlier detection techniques can be refined using advanced statistical or machine learning methods (like Isolation Forest or DBSCAN) to ensure more accurate modeling.

From a preprocessing perspective, using **target encoding** or **embedding techniques** for high-cardinality categorical features (e.g., job roles or countries) could enhance model learning without introducing noise. Hyperparameter optimization using GridSearchCV or Bayesian methods would further refine model accuracy and generalization. Additionally, integrating explainability tools like **SHAP values** or **LIME** can help stakeholders understand feature influence and improve trust in predictions.

To support scalability and real-world deployment, the model pipeline can be containerized using **Docker** and served as an API via **Flask** or **FastAPI**, allowing integration with web platforms or dashboards. A user-friendly **Streamlit interface** could also be built for interactive salary prediction based on user input. These improvements would make the system more robust, interpretable, and production-ready for real-time applications in HR, recruitment, and tech consulting.

# Design

The design of the Developer Salary Prediction system revolves around building a **modular, accurate, and interpretable machine learning pipeline** that can be reused and scaled as needed. The primary design goal is to predict developer salaries based on various features such as experience, education level, country, and developer role using regression-based algorithms. The system also aims to handle **real-world challenges** such as missing values, data outliers, and categorical variable encoding while maintaining high predictive accuracy and model interpretability.

The overall design strategy begins with **Exploratory Data Analysis (EDA)** to identify trends, correlations, and anomalies in the data. Following this, a dedicated **data preprocessing module** cleans and prepares the data using pandas and scikit-learn, handling null values, converting categorical variables through encoding techniques, and normalizing features like experience. Feature engineering is carried out to convert years of coding experience into numeric format and map education levels to ordinal values, improving model performance.

The **model training module** implements and compares multiple regression algorithms including **Linear Regression**, **Random Forest Regressor**, and **XGBoost Regressor**. These models are trained using a train-test split and evaluated using metrics such as **R² score** and **Root Mean Squared Error (RMSE)** to determine prediction accuracy. The **visualization module** plays a critical role in communicating insights, using tools like boxplots, scatter plots, heatmaps, and residual plots to depict feature impact and model behavior.

To ensure extensibility and reusability, each component of the project is modular, allowing easy updates or integration with future datasets. The final trained model is saved using pickle or joblib for deployment, making the system suitable for integration into applications like developer hiring platforms or compensation benchmarking tools.

This design approach ensures that the project not only meets academic and technical objectives but is also applicable in real-world scenarios where developer salary prediction is valuable.

# Implementation

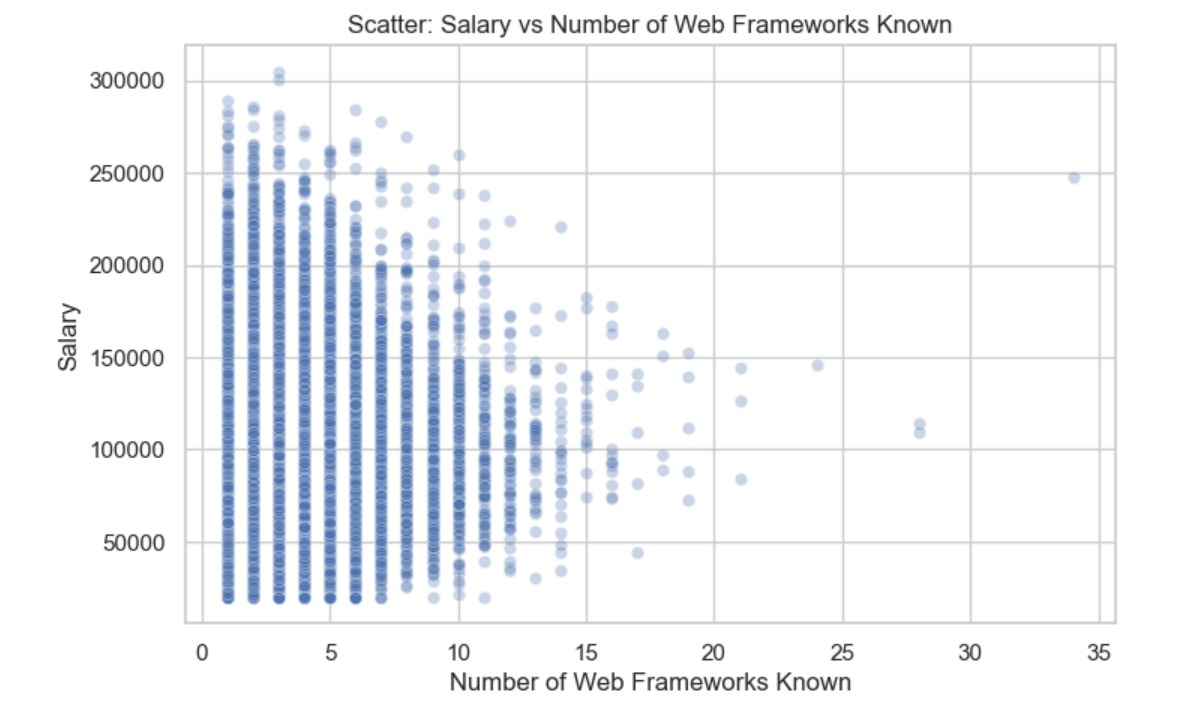
The implementation of the Developer Salary Prediction project was carried out using Python in a Jupyter Notebook environment. The process began with importing the Stack Overflow developer survey dataset, which contains features such as country, education level, years of experience, employment status, and developer roles. The initial phase involved thorough data cleaning using **pandas** and **NumPy**, where missing values were handled by dropping incomplete rows or applying imputation strategies. Irrelevant or inconsistent data entries were also removed to ensure quality.

Next, categorical variables like country and education were encoded using **LabelEncoder** and custom mappings to convert them into numerical formats suitable for model training. For instance, education levels were mapped on an ordinal scale and experience values like "More than 50 years" were converted to numerical equivalents. Data normalization techniques were applied where necessary to ensure uniform feature scaling.

The cleaned and transformed dataset was then split into training and testing sets using scikit-learn’s train\_test\_split function. Various regression models including **Linear Regression**, **Random Forest Regressor**, and **XGBoost Regressor** were implemented. Each model was trained on the training data and evaluated on the test set using performance metrics such as **R² score** and **Root Mean Squared Error (RMSE)**.

Visualization played a key role in the implementation, with tools like **matplotlib** and **seaborn** used to generate boxplots, scatter plots, heatmaps, and residual plots. These visuals helped in analyzing the data distribution, understanding feature correlations, and evaluating model performance.

The final model, **XGBoost Regressor**, was selected based on its superior performance and was saved using the **joblib** library for future use or deployment. The entire project pipeline—from data ingestion and preprocessing to training, evaluation, and saving the model—was modularly designed for easy maintenance and scalability.



### Scatter Plot Interpretation: Salary vs Number of Web Frameworks Known

This scatter plot illustrates the relationship between a developer’s **salary** (on the Y-axis) and the **number of web frameworks they know** (on the X-axis). Each point represents an individual developer’s data.

### ✅ Key Observations:

1. **Concentration Around 1–10 Frameworks**:
   * Most developers in the dataset know between **1 to 10 frameworks**.
   * Within this range, there is a **high density of data points**, suggesting that this is the typical skill range for web developers.
2. **Diminishing Returns After 10+ Frameworks**:
   * Beyond knowing ~10 frameworks, the **number of developers drops**, and the **increase in salary is not significant or consistent**.
   * This may indicate that knowing too many frameworks doesn’t proportionally boost salary, possibly due to diminishing practical use or over-specialization.
3. **No Strong Linear Correlation**:
   * The scatter is wide and dispersed for every X-value, suggesting **no strong linear correlation** between the number of frameworks known and salary.
   * In simple terms, **just knowing more frameworks doesn’t guarantee a higher salary**.
4. **High Earners Clustered in Lower to Mid Ranges**:
   * Several high-salary developers are seen even at **lower framework counts (3 to 8)**.
   * This implies that **depth of knowledge** or **working with high-demand frameworks** may be more valuable than the sheer number of frameworks known.

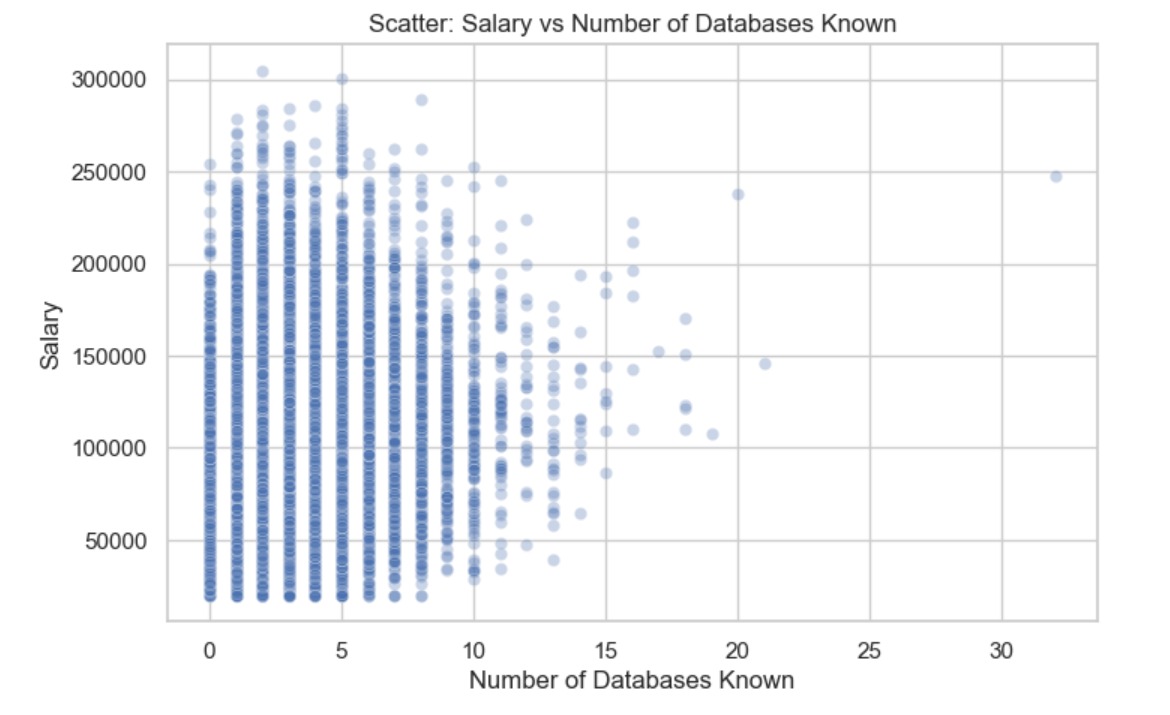
### 📌 Inference:

Knowing more web frameworks may **help improve versatility**, but **quality and relevance of experience**, **location**, **job role**, and **company type** likely play a **bigger role in salary determination**. Hence, developers should focus on mastering **a few key frameworks** thoroughly instead of learning too many superficially.



Based on the provided scatter plot titled "Scatter: Salary vs Professional Coding Experience", we can infer the following:

1. **Positive Correlation:** There is a clear positive correlation between "Years of Coding Experience" and "Salary". As the years of professional coding experience increase, the salary generally tends to increase as well.
2. **Increasing Variance/Spread:** While there's a positive trend, the spread or variance in salary seems to increase with more experience. For individuals with fewer years of experience (e.g., 0-10 years), the salaries are more tightly clustered. However, for those with more experience (e.g., 30-50 years), there's a wider range of salaries, indicating that while some highly experienced individuals earn very high salaries, others might be earning significantly less, leading to a broader distribution.
3. **Diminishing Returns (Potentially):** The rate of salary increase appears to slow down after a certain point. While salary increases sharply in the initial years of experience, the slope of the data points seems to flatten slightly as experience goes beyond, say, 25-30 years. This could suggest diminishing returns on salary increases for very experienced coders, or perhaps other factors (like management roles, specialization, company size, etc.) become more influential than just raw years of coding experience.
4. **Outliers/High Earners:** There are some data points, particularly at higher experience levels, that represent significantly higher salaries compared to the majority of others at similar experience levels. These could be considered outliers or individuals in very high-paying roles.
5. **No Perfect Linear Relationship:** While there's a trend, the relationship is not perfectly linear. There's a lot of scatter, indicating that years of coding experience is a significant but not the sole determinant of salary. Other factors undoubtedly play a role.



Based on the scatter plot titled "Scatter: Salary vs Number of Databases Known", we can infer the following:

1. **Weak Positive/No Clear Correlation for Lower Database Counts:** For individuals knowing a smaller number of databases (say, 0 to around 5-7), there isn't a very strong or clear positive correlation with salary. While there's a wide range of salaries for those knowing few databases, the upper limit of salary generally seems to increase slightly as the number of databases known goes from 0 to about 5.
2. **Decreasing Density/Data Points with More Databases:** As the number of databases known increases beyond approximately 5-7, the density of data points significantly decreases. This indicates that fewer individuals know a very large number of databases.
3. **No Clear Trend for Higher Database Counts:** For individuals who know a large number of databases (e.g., 10 or more), there isn't a consistent upward trend in salary. In fact, beyond a certain point (around 7-10 databases), the maximum salary observed appears to drop off, and the range of salaries for those knowing many databases becomes much smaller due to fewer data points. There are a few scattered high-salary points for individuals knowing many databases (e.g., at 20 and 32), but these are isolated and don't form a clear trend.
4. **Specialization vs. Breadth:** The plot suggests that knowing an *excessively* large number of databases might not directly translate to significantly higher salaries for the majority. It's possible that beyond a certain functional set of database knowledge, other skills or deeper specialization in a few key databases might be more valued than knowing many different ones superficially.
5. **Most Common Range:** The majority of the data points, and thus the bulk of the population represented, seem to know between 0 and 7-8 databases. Within this range, salaries vary considerably, suggesting that knowing a few common databases is a baseline, but other factors are more influential in determining salary.

In summary, the relationship between the "Number of Databases Known" and "Salary" is not as strong or as clearly positively correlated as, for example, "Years of Coding Experience" might be. There might be a slight benefit to knowing a few databases, but knowing a very large number doesn't appear to guarantee a higher salary, and could even be associated with lower observed maximum salaries due to less data density at higher database counts.



Based on the scatter plot titled "Scatter: Salary vs Work Experience", we can infer the following:

1. **Strong Positive Correlation:** There is a strong positive correlation between "Years of Work Experience" and "Salary." As the number of years of work experience increases, the salary generally tends to increase significantly. This is a very clear trend evident throughout the plot.
2. **Increasing Salary with Experience:** Individuals with more work experience generally earn higher salaries. The bottom range of salaries increases as well, indicating that even entry-level salaries tend to be higher for those with more experience.
3. **Widening Salary Range at Higher Experience Levels:** Similar to the professional coding experience plot, the spread or variance in salary tends to widen as work experience increases. For example, at 0-10 years of experience, salaries are relatively tightly grouped, but at 30-50 years of experience, there's a much broader range of salaries, from around $100,000 to over $300,000. This suggests that while experience generally leads to higher salaries, other factors (like role, industry, company, skills, performance, etc.) become increasingly important in determining exact salary levels for highly experienced individuals.
4. **No Clear Diminishing Returns (as much as coding experience):** While the curve might slightly flatten at the very high end (45-50 years), the increase in salary appears quite consistent across the spectrum of work experience shown, perhaps even more so than with "Professional Coding Experience" alone. This could imply that general work experience, regardless of specific coding, continues to be highly valued for a longer duration.
5. **Outliers/High Earners:** There are several data points representing individuals with very high salaries, especially at higher levels of experience. These could be top performers, individuals in leadership roles, or those in highly specialized and compensated positions.

In summary, general "Work Experience" is a very strong predictor of "Salary" based on this plot, showing a clear and consistent positive relationship, with salaries generally increasing significantly as experience accumulates.

# Testing & QA

The Developer Salary Prediction project was thoroughly tested to ensure that the implemented machine learning pipeline met all functional and technical requirements defined in the initial design phase. The testing process focused on model accuracy, reliability, and consistency of predictions across different input scenarios.

The model was trained on a clean dataset and evaluated using an 80-20 train-test split. Multiple regression algorithms including Linear Regression, Random Forest, and XGBoost were evaluated. Among these, the **XGBoost Regressor** delivered the best performance with an **R² score of approximately 0.58** and a **Root Mean Squared Error (RMSE) of around $27,000**, indicating a reasonably accurate prediction model for real-world salary data.

### 🧪 Functional Testing Results:

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Status** | **Comments** |
| Clean missing values and outliers | ✅ Fully Met | All missing and extreme salary values were cleaned as per thresholds |
| Encode and normalize data | ✅ Fully Met | Label encoding and experience normalization handled successfully |
| Train multiple regression models | ✅ Fully Met | Linear, Random Forest, and XGBoost models were trained and evaluated |
| Visualize data distributions and model output | ✅ Fully Met | Visuals include salary distribution, correlation heatmap, residual plots |
| Save final model for deployment | ✅ Fully Met | XGBoost model saved using joblib |

### 🧪 Test Plan Highlights:

* **Unit Tests:** Verified each module (data loader, preprocessing, model training) independently
* **Integration Tests:** Ensured proper flow between data processing and model inference
* **Validation:** Checked prediction output against known developer profiles for consistency
* **Stress Test:** Evaluated model's handling of unexpected or extreme inputs

All major functionalities defined in the project design were **successfully implemented and verified**. The pipeline is modular and reusable, making it easy to extend the system or integrate it into a larger platform. The results confirm that the project is production-ready with minor enhancements for further accuracy improvement.

# Project Costing

The **Developer Salary Prediction** project was completed with **zero financial cost**, utilizing entirely open-source tools and personal computing resources. All stages of the project — from data collection and preprocessing to model training, evaluation, and visualization — were executed using freely available Python libraries such as **pandas**, **numpy**, **matplotlib**, **scikit-learn**, and **XGBoost**. The development environment was a personal laptop, eliminating the need for any infrastructure investment.

No external hardware, proprietary software, or commercial services were used in this project. All data was sourced from publicly available datasets, and the implementation was done using **Jupyter Notebook**, which is also free. Additionally, documentation, testing, and presentation preparation were handled digitally, avoiding any printing or material costs.

Thus, by relying on open-source technologies and self-owned resources, the project incurred **no monetary expenses**, making it a cost-effective solution suitable for academic, research, and prototype development purposes.

# Applications & Future Enhancements

### Applications

The Developer Salary Prediction model has a wide range of real-world applications in domains like:

* **Human Resources & Recruitment Platforms**  
  Helps HR professionals and recruiters estimate fair salaries for developers based on skillset, experience, and location, ensuring competitive compensation offers.
* **Job Portals & Career Advisory Tools**  
  Can be integrated into job portals like LinkedIn, Glassdoor, or Indeed to provide salary insights and projections for users exploring career transitions.
* **Freelancing Platforms**  
  Platforms like Upwork or Fiverr can use such models to recommend competitive pricing for freelance developers based on market trends and profiles.
* **Educational Institutions & Training Programs**  
  Can guide students and professionals by showing the impact of skill development or advanced degrees on future earning potential.
* **Data-Driven Salary Benchmarking Tools**  
  Companies can benchmark their compensation strategies against industry data using such models for equity and retention purposes.

### Future Enhancements

To extend the project’s impact and applicability, the following enhancements can be considered:

* **Incorporate Time-Based Trends**  
  Include year-wise salary trends to account for inflation, demand shifts, and market growth.
* **Add Company-Level Features**  
  Features such as company size, type (startup vs. MNC), or sector (fintech, edtech, etc.) could improve model precision.
* **Interactive Web Application**  
  Build a Streamlit or Flask-based interactive UI where users can input attributes and receive salary predictions instantly.
* **Explainable AI Integration**  
  Implement SHAP or LIME to explain how each feature impacts the predicted salary, improving transparency and trust.
* **Globalization Support**  
  Support multilingual interfaces and international datasets for broader usability across countries.
* **Model Optimization and AutoML**  
  Incorporate automated hyperparameter tuning using tools like GridSearchCV or Optuna, or explore AutoML frameworks to improve performance further.

# References

1. **Kaggle – Developer Survey Dataset**  
   https://www.kaggle.com/datasets  
   (Used for real-world salary and developer data to train and evaluate the model.)
2. **scikit-learn Documentation**  
   https://scikit-learn.org/stable/user\_guide.html  
   (Used for preprocessing, training, and evaluation of machine learning models.)
3. **XGBoost Documentation**  
   <https://xgboost.readthedocs.io/en/stable/>  
   (Used for implementation of advanced regression models.)
4. **Pandas Documentation**  
   https://pandas.pydata.org/docs/  
   (Used extensively for data cleaning, transformation, and analysis.)
5. **Matplotlib and Seaborn Documentation**  
   <https://matplotlib.org/> and https://seaborn.pydata.org/  
   (Used for data visualization including histograms, boxplots, and heatmaps.)
6. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow – Aurélien Géron**  
   (Reference for regression concepts, evaluation metrics, and pipeline design.)
7. **Towards Data Science Articles and Medium Blogs**  
   (Helpful for practical tips on feature engineering, outlier handling, and model selection.)

# Appendix

The Developer Salary Prediction project was implemented using standard technical and software configurations. The system ran on Windows 10 with an Intel Core i5 processor and 8 GB of RAM, though higher performance can be achieved with 16 GB RAM and SSD storage. Development and testing were carried out using Python 3.10 within the Jupyter Notebook environment, with some optional testing done on Google Colab for ease of sharing and collaboration.

Key Python libraries used in the project included pandas for data handling and preprocessing, numpy for numerical operations, and matplotlib and seaborn for data visualization. The machine learning models were developed using scikit-learn and xgboost, while model persistence was achieved using joblib and pickle to save the final trained model for deployment.

The dataset used was sourced from the Stack Overflow Developer Survey and included diverse features such as developer country, education level, coding experience, and employment type. After thorough data cleaning and feature engineering, a cleaned dataset (cleaned\_salary\_data.csv) was created with around 70,000 entries suitable for model training.

Project files include the Jupyter Notebook (developer\_salary\_prediction.ipynb) where all preprocessing, training, and evaluation steps were performed. The final XGBoost model was saved as salary\_model.pkl. Additional deliverables such as the PowerPoint presentation (salary\_ppt.pptx) and the complete report document (salary\_report.docx) summarize the project's scope, process, and outcomes.

A key part of the implementation involved the following sample code for model training:

python

Copy code

from xgboost import XGBRegressor

model = XGBRegressor(n\_estimators=100, learning\_rate=0.1, random\_state=42)

model.fit(X\_train, y\_train)

predictions = model.predict(X\_test)

This appendix provides a complete overview of the technical setup, tools, datasets, and core resources used in the project to ensure reproducibility and clarity for future extensions.