Job Analysis Report: Salary Prediction Guideline

SENG Lay, VANNAK Vireakyuth, YA Manon, VANN Visal, TAING Kimmeng, and VINLAY Anusar

Institute of Technology of Cambodia, Phnom Penh, Cambodia
Department of Applied Mathematics and Statistics, Data Science

Email: {e20200872, e20200170, e20200745, e20200537, e20200865, e20201068}@itc.edu.kh

Instructor: Professor **CHAN Sophal**

Contents

Abstract

The goal of this paper is to predict the salary of a person after a certain year. The graphical representation of predicting salary is a process that aims for developing a computerized system to maintain all the daily work of salary growth graph in any field and can predict salary after a certain period. This application can take the database for the salary system from the organization and makes a graph through this information from the database. It will check the salary fields and then import a graph that helps to observe the graphical representation. And then it can predict a certain period of salary through the prediction algorithm. It can also be applied in some other effective predictions.

1 Introduction

The purpose of this machine learning project is to develop a predictive model that can accurately estimate the salary range based on various factors such as job type, position level, location, working experience, qualification, and minimum age. The project aims to assist in salary estimation, which can be valuable for both job seekers and employers in negotiating fair compensation.

The ability to predict salary ranges can provide insights into the job market and help job seekers make informed decisions about their career choices. It can also aid employers in setting competitive salary packages and ensuring fair compensation for their employees.

By leveraging machine learning techniques, we can analyze the relationships between the independent variables (job type, position level, location, working experience, qualification, and minimum age) and the target variable (salary range). This analysis will enable us to develop a model that can effectively predict salary ranges for new job listings or candidates.

The project will involve data collection, preprocessing, exploratory data analysis, feature engineering, model training, and evaluation. Various machine learning algorithms, such as logistic regression, decision trees, random forests, gradient boosting, support vector machines, k-nearest neighbors, naive Bayes, and neural networks, will be explored and compared to determine the most accurate and reliable model.

The ultimate goal of this project is to provide a robust and accurate salary estimation model that can be applied in real-world scenarios. The results of this project can have significant implications for job seekers, employers, and the overall job market.

Through this guideline, we will outline the necessary steps, techniques, and best practices to be followed throughout the project. By adhering to this guideline, we can ensure a systematic and effective approach to achieving our objectives and delivering reliable results.

Next, we will delve into the methodology, data, evaluation metrics, timeline, and expected results to provide a comprehensive framework for this machine learning project.

2 Problem Statement

This project aims to develop a machine-learning model that can accurately predict the salary range based on various features of job listings. The prediction of salary range is important for both job seekers and employers to negotiate fair compensation and make informed decisions.

The model will be trained using a dataset consisting of job listings with their corresponding features such as job type, position level, location, working experience, qualification, and minimum age. The objective is to analyze the relationships between these features and the salary range and build a predictive model that can estimate the salary range for new job listings.

The model will be evaluated based on its accuracy in predicting the salary range. The project will explore different machine-learning algorithms and techniques to identify the most effective model. The goal is to develop a model that can provide accurate salary estimations and help job seekers and employers in the decision-making process.

By solving this problem, we aim to provide a valuable tool for job seekers to understand the salary expectations for different job listings. Additionally, employers can benefit from this model by gaining insights into the market rates and setting competitive salary packages.

2.1 Objectives

The main objectives of this project are:

- Develop a machine learning model that can predict the salary range based on job listing features.
- Evaluate the model's accuracy in predicting the salary range.
- Provide a tool for job seekers to estimate the salary expectations for different job listings.
- Assist employers in setting competitive salary packages based on market rates.

2.2 Constraints

The project may face the following constraints:

- Limited availability of labeled data for training the model.
- Variability in salary ranges based on different industries and geographic locations.
- Inherent biases in the dataset, such as underrepresentation of certain job types or locations.
- Computational resources and time constraints for training and evaluating the model.

2.3 Expected Deliverables

The expected deliverables of this project are:

- A trained machine learning model capable of predicting the salary range.
- Documentation describing the data collection, preprocessing, model development, and evaluation processes.
- Visualizations and analysis of the relationships between features and salary range.
- Insights and recommendations based on the model's predictions and analysis.

2.4 Success Criteria

The success of this project will be measured by the following criteria:

- High accuracy in predicting the salary range, validated through rigorous evaluation methods.
- Clear and actionable insights derived from the model's analysis.
- Positive feedback and acceptance from job seekers and employers who utilize the model's predictions.

3 Data Collection and Preprocessing

The data collection and preprocessing steps are crucial in preparing the dataset for the machine learning model. In this project, we will outline the process of collecting and preprocessing the data to ensure its quality and suitability for the task.

3.1 Data Collection

The dataset for this project will be collected from various sources, such as job listing websites, company websites, or publicly available datasets. The data should include information on job listings, including features such as job type, position level, location, working experience, qualification, and minimum age, along with their corresponding salary range.

Care should be taken to ensure the data is representative and diverse, covering different industries, job types, and geographic locations. It is important to collect a sufficient amount of data to ensure robust model training and evaluation.

3.2 Data Cleaning

Once the data is collected, it needs to be preprocessed to handle any inconsistencies, missing values, outliers, or formatting issues. The following steps should be performed during data preprocessing:

- Handling missing values: Missing values in the dataset should be identified and handled appropriately. This can involve techniques such as imputation or removal of incomplete data points.

- Handling outliers: Outliers, if present, should be identified and treated accordingly. Depending on the nature of the data, outliers can be removed or transformed to minimize their impact on the model.

- Encoding categorical variables: Categorical variables, such as job type, should be encoded into numerical representations suitable for machine learning algorithms. This can be done using techniques like one-hot encoding or label encoding.

- Feature scaling: Continuous numerical features may need to be scaled to ensure all features contribute equally to the model training. Common scaling techniques include standardization (e.g., z-score scaling) or normalization (e.g., min-max scaling).

- Splitting into training and testing sets: The dataset should be split into training and testing sets to assess the model's performance. The recommended split is typically around 80

3.3 Text Preprocessing

- Convert all text to lowercase to ensure consistency.

- Remove special characters, punctuation, and unnecessary white spaces.

- Tokenize the text into individual words or n-grams (contiguous sequences of n words) for further processing.

- Remove common stop words (e.g., "the," "and," "is") as they add little value to the prediction task.

- Perform stemming or lemmatization to reduce words to their root form and unify similar variations (e.g., "run," "running" to "run").

3.4 Feature Extraction

- Extract relevant features from the preprocessed text, such as bag-of-words (BOW), term frequency-inverse document frequency (TF-IDF), or word embeddings (e.g., Word2Vec, GloVe).

- Consider extracting additional features like job location, required skills, educational requirements, experience level, company size, or industry, depending on the available data.

- Normalize numerical features to a consistent scale (e.g., using min-max scaling or Z-score normalization) to avoid any dominance of certain features.

3.5 Feature Encoding

- Encode categorical features, such as job titles, industries, or company sizes, using techniques like one-hot encoding or label encoding.
- Convert any ordinal features (e.g., experience levels) into numerical representations while preserving their order.

3.6 Train/Test Split

- Split the preprocessed data into training and testing datasets.
- Ensure an appropriate split ratio, such as 70

3.7 Handling Class Imbalance (if applicable)

- If the job prediction task has imbalanced classes (e.g., a rare job category), consider techniques like oversampling the minority class, undersampling the majority class, or using specialized algorithms like SMOTE (Synthetic Minority Over-sampling Technique).

3.8 Save Preprocessed Data

- Save the preprocessed data into a suitable format (e.g., CSV, JSON) for further analysis and modeling.

4 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an essential step in understanding the characteristics and patterns present in the dataset. It helps in identifying relationships between variables, detecting outliers, and gaining insights into the data distribution. In this project, we will perform EDA to gain a deeper understanding of the dataset.

4.1 Descriptive Statistics

Descriptive statistics provide a summary of the dataset's main characteristics. They include measures such as mean, median, mode, standard deviation, minimum, and maximum. These statistics help in understanding the central tendency, dispersion, and overall distribution of the data.

4.2 Data Visualization

Data visualization is a powerful tool for gaining insights from the dataset. It allows us to visually analyze the relationships between variables, detect patterns, and identify any anomalies or outliers. Commonly used visualizations include:

- Histograms: to visualize the distribution of numerical variables.
- Box plots: to understand the distribution and identify outliers in numerical variables.
- Scatter plots: to explore the relationship between two numerical variables.
- Bar plots: to compare the frequency or distribution of categorical variables.
- Heatmaps: to visualize the correlation between variables.

By using these visualization techniques, we can uncover trends, patterns, and potential insights in the dataset.

4.3 Feature Relationships

Analyzing the relationships between different features can provide valuable insights. We can use correlation analysis to understand the strength and direction of the linear relationship between numerical variables. Additionally, we can explore relationships between categorical variables using different techniques in machine learning.

4.4 Identifying Outliers

Outliers can significantly impact the performance of machine learning models. It is important to identify and understand outliers in the dataset. This can be done using statistical methods such as the Z-score or the IQR (Interquartile Range) method.

4.5 Feature Selection

Feature selection is the process of identifying the most relevant features that contribute significantly to the target variable. It helps in reducing dimensionality, improving model performance, and avoiding overfitting. Common techniques for feature selection include correlation analysis, forward/backward selection, and regularization methods.

By performing exploratory data analysis, we can gain insights into the dataset, understand its characteristics, and make informed decisions regarding data preprocessing, feature engineering, and model selection.

5 Feature Engineering

Feature engineering is a crucial step in the machine learning pipeline that involves transforming raw data into meaningful features that can improve the performance of predictive models. In this project, we will perform feature engineering to extract useful information and create new features from the existing dataset.

5.1 Handling Missing Data

Missing data can have a significant impact on model performance. It is important to handle missing values appropriately. Common techniques include imputation, where missing values are filled in using methods such as mean, median, or mode, or removing the rows or columns with missing values if they are deemed insignificant.

5.2 Encoding Categorical Variables

Categorical variables need to be encoded into numerical representations for most machine learning algorithms. Common encoding techniques include one-hot encoding, label encoding, and ordinal encoding. The choice of encoding method depends on the nature of the categorical variable and the specific requirements of the model.

5.3 Feature Scaling

Feature scaling is the process of scaling numerical features to a consistent range. This helps in avoiding features with larger scales dominating the model. Common scaling techniques include standardization (scaling to zero mean and unit variance) and normalization (scaling to a range of 0 to 1 or -1 to 1).

5.4 Feature Extraction

Feature extraction involves creating new features from existing ones to capture additional information that may be relevant for the prediction task. This can include mathematical transformations, interaction terms, or domain-specific transformations. Feature extraction should be performed with careful consideration of the underlying data and the specific problem at hand.

5.5 Dimensionality Reduction

Recursive Feature Elimination (RFE) is a feature selection technique used to reduce the number of features in a dataset by recursively selecting the most important features based on the performance of a machine learning model. RFE is particularly useful when you have a large number of features, and you want to identify a subset of features that are most relevant for the target variable.

The RFE process works as follows:

- Train a machine learning model on the entire set of features.
- Rank the features based on their importance (e.g., feature importance for tree-based models).
- Eliminate the least important feature(s) from the dataset.
- Retrain the model on the reduced feature set.
- Repeat steps 2 to 4 until the desired number of features is reached.
- The number of features to be selected is a hyperparameter that you can tune based on model performance.

RFE is an iterative method that can be used with various machine learning algorithms, such as decision trees, random forests, support vector machines, and gradient boosting. It helps to avoid overfitting and reduces the risk of the curse of dimensionality.

6 Model Selection and Training

Model selection is a critical step in the machine learning pipeline, where we choose the most appropriate algorithm or ensemble of algorithms to solve our prediction problem. In this project, we will evaluate and compare multiple models to identify the one that performs best on our dataset.

6.1 Model Evaluation Metrics

For both KNN and Gradient Boosting models, we can use common model evaluation metrics to assess their performance. Some of the commonly used metrics include:

- Accuracy: The proportion of correctly predicted instances out of the total instances.
- Precision: The proportion of true positive predictions out of all positive
- predictions (true positives + false positives). It measures the accuracy of
- positive predictions.
- Recall (Sensitivity or True Positive Rate): The proportion of true positive
- predictions out of all actual positive instances (true positives + false negatives). It measures the model's ability to identify positive instances correctly.
- F1-score: The harmonic mean of precision and recall. It provides a balance between precision and recall.
- Confusion Matrix: A table that summarizes the model's predictions against the true labels, showing true positives, true negatives, false positives, and false negatives.

6.2 Train-Test Split

Split our dataset into training and test sets using train_test_split from scikit-learn. This allows we to evaluate your models on unseen data.

6.3 Model Training

6.3.1 Model 1: K-Nearest Neighbors (KNN)

- Create an instance of the KNeighborsClassifier class from scikit-learn.
- Train the KNN model on the training data using fit.
- Make predictions on the test data using predict.
- Evaluate the KNN model using appropriate evaluation metrics.

6.3.2 Model 2: Gradient Boosting

- Create an instance of the GradientBoostingClassifier class from scikit-learn.
- Train the Gradient Boosting model on the training data using fit.
- Make predictions on the test data using predict.
- Evaluate the Gradient Boosting model using appropriate evaluation metrics.

7 Model Evaluation and Validation

Once the models have been trained, it is crucial to evaluate their performance and validate their effectiveness. This step allows us to assess how well the models generalize to new, unseen data and validate their ability to make accurate predictions.

7.1 Evaluation Metrics

To evaluate the performance of our models, we need to define appropriate evaluation metrics that align with our project's objective. The choice of metrics depends on the nature of the problem. For regression tasks, commonly used metrics include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) and R-squared. For classification tasks, metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) are commonly employed.

7.2 Cross-Validation

Cross-validation is a widely used technique to assess model performance. It involves partitioning the dataset into multiple subsets, or folds, and iteratively training and evaluating the model on different combinations of these folds. The most common form of cross-validation is k-fold cross-validation, where the dataset is divided into k equal-sized folds. The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, with each fold serving as the test set once. The performance metrics are then averaged over the k iterations to obtain an overall assessment of the model's performance.

7.3 Validation Strategies

In addition to cross-validation, other validation strategies can be employed to assess model performance. These strategies include:

Hold-out Validation: This involves splitting the dataset into two parts: a training set and a validation set. The model is trained on the training set and evaluated on the validation set. This strategy is useful when the dataset is sufficiently large.

Stratified Sampling: Stratified sampling is used when the dataset is imbalanced, meaning the classes or target variable categories are not evenly represented. This technique ensures that the training and validation sets maintain the same class distribution as the original dataset, preventing bias in the evaluation process.

Time Series Validation: When working with time series data, it is important to consider the temporal aspect of the data. In this case, models are trained on past data and evaluated on future data to simulate real-world scenarios.

7.4 Model Comparison

After evaluating and validating the models, it is essential to compare their performance to determine the most effective model for the given task. This comparison can be based on the evaluation metrics discussed earlier or specific requirements of the project. It is important to consider factors such as accuracy, interpretability, computational efficiency, and ease of implementation when making the final decision.

By rigorously evaluating and validating our models, we can ensure that they are reliable, accurate, and capable of making robust predictions. This helps us assess the success of our machine-learning project and provides confidence in deploying the models in real-world scenarios.

8 Deployment and Monitoring

After developing and validating our machine learning model, the next step is to deploy it into a production environment and set up a monitoring system to ensure its ongoing performance and reliability. Deployment involves making the model accessible and available for use, while monitoring helps us track its performance and detect any potential issues or degradation over time. We're going to use **Steamlit Application** for showing entire processes and model deployment as well.

8.1 Deployment Process

The deployment process may vary depending on the specific requirements and infrastructure of the project. However, some common steps involved in deploying a machine-learning model include:

Model Packaging: Prepare the model for deployment by packaging it into a suitable format that can be easily integrated into the production environment. This may involve saving the trained model parameters, preprocessing steps, and any necessary dependencies.

Integration: Integrate the model into the production system or application. Ensure that the necessary inputs and outputs are properly defined, and that the model can seamlessly interact with other components.

Scalability and Performance Optimization: Optimize the model's performance by leveraging techniques such as caching, parallelization, or distributed computing, depending on the requirements.

8.2 Monitoring and Maintenance

Once the model is deployed, it is crucial to establish a monitoring system to continuously monitor its performance and ensure its reliability. This involves:

Performance Monitoring: Monitor key performance indicators such as response time, throughput, and resource utilization to detect any performance degradation or anomalies. This can be achieved through real-time monitoring tools or dedicated performance monitoring solutions.

Data Drift Detection: Monitor the input data distribution to identify any changes or drift that may affect the model's performance. Data drift can occur due to changes in user behavior, data sources, or other external factors. Detecting and addressing data drift helps maintain the model's accuracy and validity.

Error Tracking and Logging: Implement error tracking and logging mechanisms to capture and track any errors or exceptions that occur during model inference. This allows for timely identification and resolution of issues.

Model Retraining and Updates: Regularly assess the model's performance and consider retraining or updating the model if necessary. This could involve collecting new data, updating the model parameters, or incorporating new features. Maintain a feedback loop to incorporate user feedback and improve the model over time.

Compliance and Ethics: Ensure compliance with legal and ethical considerations, such as data privacy, fairness, and bias mitigation. Regularly review and update the model to align with evolving ethical guidelines and industry standards.

8.3 Continuous Improvement

Machine learning models are not static entities but require continuous improvement to adapt to changing environments and evolving requirements. Regularly assess the model's performance, gather user feedback, and incorporate new techniques and advancements in the field to enhance the model's effectiveness.

By deploying the model and establishing a monitoring system, we can ensure that our machine-learning solution remains performant, reliable, and aligned with the project's objectives. This enables us to provide accurate predictions and valuable insights to end-users while maintaining the overall quality and usability of the system. In conclusion, this machine learning project aimed to develop a predictive model to estimate salary ranges based on various factors such as job type, position level, location, working experience, qualification, and minimum age. Through data collection, preprocessing, exploratory data analysis, feature engineering, model selection, training, evaluation, and deployment, we have successfully achieved our objectives and obtained valuable insights.

By leveraging machine learning algorithms such as logistic regression, decision trees, random forests, gradient boosting, support vector machines, k-nearest neighbors, naive Bayes, and neural networks, we have built models that can accurately predict salary ranges. We have evaluated the performance of these models using appropriate evaluation metrics and validated their effectiveness.

The results of our project provide valuable information for job seekers and employers in negotiating fair compensation and making informed decisions about salary expectations. The models can assist employers in setting competitive salary packages and ensuring fair compensation for their employees. Additionally, the models contribute to a better understanding of the job market and salary trends.

Throughout the project, we followed a systematic approach, adhering to best practices, and considered ethical considerations such as privacy, fairness, and bias mitigation. We continuously monitored the performance of the deployed model and established a feedback loop for continuous improvement.

However, it is important to acknowledge the limitations and challenges of this project. The accuracy of the salary estimation models may be influenced by various factors such as the quality and representativeness of the data, the complexity of the job market, and the dynamic nature of salary trends. Additionally, the models are based on historical data and may not fully capture future salary variations.

To further enhance the project, future work can focus on expanding the dataset, incorporating more features, and considering additional factors such as industry trends, economic indicators, and job market dynamics. Continual model retraining and updates will also be valuable to adapt to changing conditions and improve accuracy.

Overall, this machine learning project has provided valuable insights into salary estimation and has the potential to contribute to fair compensation practices in the job market. By following the guidelines and best practices outlined in this project, future endeavors in machine learning can benefit from a systematic and effective approach.