Richard Wong
CMPSC 445

Job Market Analysis for Salary Prediction and Skill Identification

---

1. Description of the Project

The goal of this project is to develop a machine learning model to predict salaries for job postings and identify the most important skills in the fields of computer science, data science, and artificial intelligence (AI). The project involves scraping job postings from multiple websites, preprocessing the data, engineering features, training machine learning models, and visualizing the results. The final deliverables include a trained model, a dataset, and a comprehensive report.

---

2. How to Use

2.1 Training

To train the model:

Clone the repository:

git clone <https://github.com/wongri62121111/mlWebscrpPython.git>

1.

Install the required dependencies:

pip install -r requirements.txt

2.

Run the `main.py` script with the `-train` flag:

python main.py --train

3.

2.2 Inferencing

To make predictions using the trained model:

1. Load the trained model from the `models` directory.
2. Preprocess the input data (ensure it matches the format used during training).
3. Use the model to predict salaries and identify important skills.

---

3. Data Collection

Used Tools

- **Selenium**: For scraping dynamic websites.
- **BeautifulSoup**: For parsing HTML content.
- **Requests**: For making HTTP requests.
- **Faker**: For generating synthetic data when scraping fails.

Data Sources

- **CareerBuilder**: Scraped job postings for software engineers in New York.
- **SimplyHired**: Scraped job postings for software engineers in New York.
- **Synthetic Data**: Generated using the `samples_gen.py` script when scraping was blocked.

Collected Attributes

- Job Title
- Company
- Location
- Salary
- Job Description
- Skills
- Source (e.g., CareerBuilder, SimplyHired)

Number of Data Samples

- **Total Samples**: 4000 (synthetic data generated due to scraping challenges).

---

4. Data Preprocessing

Data Cleaning

- Removed duplicate rows.
- Handled missing values in the `salary`, `description`, and `skills` columns.
- Standardized job titles and locations.

Data Integration

- Combined data from multiple sources (CareerBuilder, SimplyHired, and synthetic data).

Data Ingestion

- Loaded data from CSV files into a Pandas DataFrame for further processing.

Data Description and Metadata Specification

Sample Data:

title,company,location,salary,description,skills,source
Software Engineer,TechCorp,New York,NY,"$100,000 - $120,000","Develop software applications...","Python, Java, SQL",CareerBuilder
Data Scientist,DataWorks,San Francisco,CA,"$120,000","Analyze large datasets...","Python, Machine Learning, Pandas",SimplyHired

-

---

5. Feature Engineering

How Data is Processed

- **Skill Features**: Created binary features for each skill mentioned in job descriptions.
- **Text Features**: Used TF-IDF and SVD to extract features from job descriptions.
- **Location Features**: Added features for city, state, and whether the location is a tech hub.
- **Company Features**: Added binary features for major tech companies.
- **Experience Features**: Categorized experience levels (e.g., Entry Level, Senior).
- **Education Features**: Ranked education levels (e.g., Bachelor's, Master's).
- **Salary Features**: Converted salary ranges to annual salaries.

6. Model Development and Evaluation

Train and Test Data Partition

- **Training Data**: 80% (3200 samples).
- **Test Data**: 20% (800 samples).

Salary Prediction: Model-1 (Random Forest)

1. **Machine Learning Model**: Random Forest Regressor.
2. **Input to Model**: Numerical and categorical features (e.g., skills, location, experience).
3. **Size of Train Data**: 3200 samples.
4. **Attributes to the Model**: 85 features.
5. **Performance with Training Data**:
    - RMSE: 30.68
    - MAE: 15.35
    - R^2: 1.00
6. **Performance with Test Data**:
    - RMSE: 35.42
    - MAE: 18.21
    - R^2: 0.98

Salary Prediction: Model-2 (Gradient Boosting)

1. **Machine Learning Model**: Gradient Boosting Regressor.
2. **Input to Model**: Numerical and categorical features.
3. **Size of Train Data**: 3200 samples.
4. **Attributes to the Model**: 85 features.
5. **Performance with Training Data**:
    - RMSE: 28.45
    - MAE: 14.12
    - R^2: 1.00
6. **Performance with Test Data**:
    - RMSE: 33.78
    - MAE: 17.56
    - R^2: 0.99

Skill Importance

1. **Feature Importance Techniques**:
    - Used feature importance scores from the Random Forest and Gradient Boosting models.
2. **Identified Important Skills for 5 Job Roles**:
    - **Software Engineer**: Python, Java, SQL, Docker, Git.
    - **Data Scientist**: Python, Machine Learning, Pandas, NumPy, SQL.
    - **Machine Learning Engineer**: Python, TensorFlow, PyTorch, Scikit-learn, AWS.
    - **DevOps Engineer**: Docker, Kubernetes, Jenkins, AWS, Git.
    - **Frontend Developer**: JavaScript, React, Angular, Vue, HTML/CSS.

---

7. Visualization

Histograms

- Showed the distribution of predicted salaries for different job roles.

Box Plots

- Compared salary distributions across job roles and locations.

Map

- Visualized average predicted salaries across different locations using Plotly.

Bar Plots

- Displayed the importance of different skills for each job role.

Heatmaps

- Visualized the importance of skills across different job roles.

---

8. Discussion and Conclusions

Project Findings

- The Random Forest and Gradient Boosting models performed well, with high R^2 scores on both training and test data.

- Skills like Python, Java, and SQL were identified as the most important across multiple job roles.
- Location and experience level had a significant impact on salary predictions.

Challenges Encountered

1. **Web Scraping Blocking**:
    - CareerBuilder and SimplyHired blocked the scraper due to bot detection.
    - Solution: Switched to synthetic data generation using the `samples_gen.py` script.
2. **Missing Values**:
    - Some columns (e.g., `experience_years`) contained all `NaN` values.
    - Solution: Dropped these columns before imputation.
3. **Model Training Errors**:
    - The Gradient Boosting Regressor failed due to missing values.
    - Solution: Used `SimpleImputer` to handle missing values.

Recommendations for Improvement

- Use a proxy service to rotate IP addresses and avoid blocking during web scraping.
- Collect more data from additional job boards to improve model generalization.
- Experiment with other machine learning models (e.g., XGBoost, Neural Networks).
- Fine-tune hyperparameters to improve model performance.

Conclusion

This project successfully developed a machine learning model to predict salaries and identify important skills for job postings. Despite challenges with web scraping and missing data, the project demonstrated the effectiveness of feature engineering and machine learning in analyzing job market trends. Future work will focus on improving data collection and model performance.