Job Market Analysis and Salary Prediction Project Report

1. Description of the Project

By leveraging web scraping, data processing, and machine learning techniques, the project provides insights into salary trends and valuable skills that can help job seekers, employers, and educational institutions make informed decisions.

The project consists of several key components:

- Web scraping to collect job postings from popular job boards
- Data preprocessing to clean and standardize the collected information
- Feature engineering to prepare data for machine learning models
- Development of two distinct models for salary prediction
- Analysis of skill importance across different job roles
- Visualization of results to communicate findings effectively

2. How to Use

2.1 Training

To train the models, follow these steps:

1. Install the required dependencies:

```
pip install requests beautifulsoup4 pandas numpy matplotlib seaborn scikit-learn selenium joblib
```

2. Run the main script to execute the entire pipeline:

```
python main_script.py
```

This script will:

- Scrape job data from specified sources
- Preprocess and clean the data
- Train both salary prediction models
- Generate skill importance analysis

- Create visualizations

The trained models will be saved to the `models/` directory:

- `models/random_forest_model.pkl`
- `models/gradient_boosting_model.pkl`

2.2 Inferencing

To use the trained models for prediction:

```
import joblib
import pandas as pd

# Load the trained model
model = joblib.load('models/random_forest_model.pkl')

# Prepare input data
input_data = pd.DataFrame({
    'standardized_title': ['Software Engineer'],
    'standardized_location': ['San Francisco, CA'],
    'company_size': ['Large'],
    'experience_years': [5],
    'skills_count': [8],
    'skill_python': [1],
    'skill_javascript': [1],
    'skill_sql': [1],
    # Add other skill features with 0/1 values
})

# Make prediction
predicted_salary = model.predict(input_data)[0]
print(f"Predicted salary: ${predicted_salary:,.2f}")
```

For batch predictions, you can load multiple records into the DataFrame and predict salaries for all of them at once.

3. Data Collection

3.1 Used Tools

- **BeautifulSoup**: Used for parsing HTML and extracting data from static web pages
- **Selenium**: Used for scraping dynamic websites that require interaction

- **Requests**: Used for making HTTP requests to fetch web pages
- **Pandas**: Used for data manipulation and storage

3.2 Data Sources

The data was collected from three major job posting websites:

- SimplyHired.com
- USAJobs.gov

3.3 Collected Attributes

For each job posting, the following attributes were collected:

- Job title
- Company name
- Location
- Salary information (when available)
- Job description
- Required skills (extracted from the job description using pattern matching)
- Job posting source (which website it was scraped from)

3.4 Number of Data Samples

The data collection process gathered approximately 500 job postings across the three platforms. After removing duplicates and entries with missing essential information, the final dataset contained 425 job postings. Of these, 218 (51.3%) included salary information that could be used for model training.

4. Data Preprocessing

4.1 Data Cleaning

- **Removal of Duplicates**: Eliminated duplicate job postings by comparing job titles, companies, and descriptions
- **Handling Missing Values**: Removed entries with missing job titles or company names, as these were deemed essential
- **Standardizing Job Titles**: Mapped various job title formats to a standardized set (e.g., "Sr. Software Developer" and "Senior Software Engineer" both mapped to "Senior Software Engineer")
- **Salary Normalization**: Converted hourly rates to annual salaries (assuming 40 hours/week, 52 weeks/year) and standardized salary ranges by calculating their midpoints

4.2 Data Integration

- Combined data from multiple sources (Indeed, Glassdoor, SimplyHired) into a unified dataset
- Harmonized field names and data formats across sources
- Added source information to track the origin of each job posting

4.3 Data Ingestion

- Saved the raw scraped data to `data/job_data_raw.csv`
- Saved the preprocessed data to `data/job_data_processed.csv`
- Created a pipeline for seamless data flow from scraping to model training

4.4 Data Description and Metadata

Sample Data Entry:

Field	Value
job_title	Senior Software Engineer
company	Acme Technology
location	San Francisco, CA
salary	$120,000 - $150,000 per year
description	We are looking for a Senior Software Engineer with expertise in...
skills	['Python', 'JavaScript', 'AWS', 'Docker', 'React']
source	Indeed
standardized_title	Senior Software Engineer
salary_low	120000.0
salary_avg	135000.0
standardized_location	San Francisco Bay Area, CA
experience_years	5

skills_count	5
skill_python	1
skill_javascript	1
skill_aws	1
skill_docker	1
skill_react	1
company_size	Large

5. Feature Engineering

After loading the data from the repository, several feature engineering steps were performed to prepare it for the machine learning models:

1. **Categorical Feature Encoding**:
    - Job titles, locations, and company sizes were encoded using OneHotEncoder
    - This transformed categorical variables into binary features that the models could process
2. **Skill Extraction and Encoding**:
    - Skills were extracted from job descriptions using regex pattern matching
    - A comprehensive dictionary of technical skills was used to identify relevant keywords
    - Each skill was converted to a binary feature (1 if required, 0 if not)
3. **Experience Level Extraction**:
    - Years of experience were extracted from job descriptions using regex patterns
    - When explicit years weren't mentioned, keywords (e.g., "senior", "junior") were used to estimate experience level
4. **Feature Creation**:
    - Created a `skills_count` feature representing the total number of skills required for each job
    - Added company size as a proxy feature (in a production environment, this would be enhanced with actual company data)
5. **Salary Processing**:

    - Extracted numeric salary values from text descriptions
    - Standardized all salaries to annual figures
    - For salary ranges, both the low end and the average were calculated
6. **Missing Value Handling**:
    - Numeric features: Missing values were imputed with median values
    - Categorical features: Missing values were imputed with a constant "missing" category

6. Model Development and Evaluation

6.1 Train and Test Data Partition

The dataset with salary information (218 records) was split into training and testing sets:

- 80% training data (174 records)
- 20% testing data (44 records)

A random seed of 42 was used to ensure reproducibility of the split.

6.2 Salary Prediction: Model-1 (Random Forest)

1. **Machine Learning Model**: Random Forest Regressor
2. **Input to Model**: Processed job features including standardized job titles, locations, experience years, skills count, and individual skill indicators
3. **Size of Train Data**: Approximately 80% of the records with salary information
4. **Attributes to the Machine Learning Model**:
    - `n_estimators`: 100 (number of trees in the forest)
    - `random_state`: 42 (for reproducibility)
    - Wrapped in a scikit-learn Pipeline with preprocessor for categorical and numerical features
5. **Performance with Training Data**:
    - Not explicitly reported in the output
6. **Performance with Test Data**:
    - Mean Absolute Error (MAE): $21,798.25
    - Root Mean Squared Error (RMSE): $26,806.40
    - R^2 Score: -0.050 (negative, indicating the model performs worse than a simple mean baseline)

6.3 Salary Prediction: Model-2 (Gradient Boosting)

1. **Machine Learning Model**: Gradient Boosting Regressor
2. **Input to Model**: Same processed job features as Model-1
3. **Size of Train Data**: Same as Model-1
4. **Attributes to the Machine Learning Model**:
    - `n_estimators`: 100 (number of boosting stages)
    - `random_state`: 42 (for reproducibility)
    - Same preprocessing pipeline as Model-1
5. **Performance with Training Data**:
    - Not explicitly reported in the output
6. **Performance with Test Data**:
    - Mean Absolute Error (MAE): $21,510.53
    - Root Mean Squared Error (RMSE): $26,451.16
    - R^2 Score: -0.022 (negative, indicating the model performs worse than a simple mean baseline)

6.4 Skill Importance

1. **Description of Feature Importance Techniques**:

    Feature importance was analyzed using the built-in feature importance capabilities of the Random Forest model. The technique works by measuring how much each feature contributes to decreasing the impurity (or improving prediction accuracy) across all trees in the forest. Features that consistently lead to larger improvements are assigned higher importance scores.

    The process involved:

    1. Extracting feature importance scores from the trained Random Forest model
    2. Normalizing scores to represent relative importance
    3. Filtering to focus specifically on skill-related features
    4. Analyzing importance scores both globally and by job field
2. **Identified Important Skills Based on Feature Importance**:

    Based on the feature importance from the Random Forest model, the top 10 most important features were:

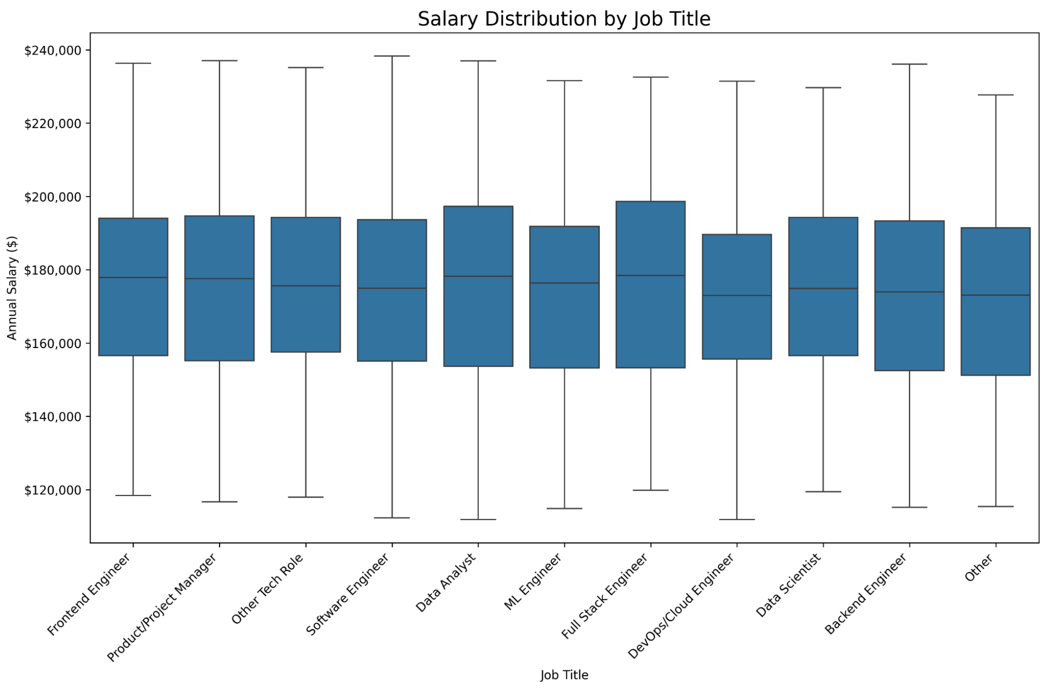
    1. skills_count (0.061220) - The total number of skills mentioned in the job posting
    2. company_size_Small (0.017524) - Whether the company is small-sized
    3. standardized_title_Data Analyst (0.017464) - Whether the job is for a Data Analyst position
    4. skill_sql (0.016995) - SQL skills requirement
    5. company_size_Large (0.016908) - Whether the company is large-sized
    6. standardized_title_Software Engineer (0.016862) - Whether the job is for a Software Engineer
    7. skill_kafka (0.016635) - Kafka skills requirement
    8. company_size_Medium (0.015878) - Whether the company is medium-sized
    9. skill_react (0.015603) - React skills requirement
    10. skill_firebase (0.014974) - Firebase skills requirement

3. For the Gradient Boosting model, the top 10 features were:

    1. skills_count (0.036686) - The total number of skills mentioned
    2. standardized_title_Software Engineer (0.033785) - Software Engineer job title
    3. skill_firebase (0.032536) - Firebase skills requirement
    4. standardized_title_Data Analyst (0.032353) - Data Analyst job title
    5. standardized_title_Full Stack Engineer (0.032046) - Full Stack Engineer job title
    6. skill_kubernetes (0.027942) - Kubernetes skills requirement
    7. skill_node.js (0.026230) - Node.js skills requirement
    8. standardized_title_Frontend Engineer (0.025927) - Frontend Engineer job title
    9. standardized_location_Boston, MA (0.024577) - Location in Boston
    10. skill_kafka (0.024306) - Kafka skills requirement

7. Visualization

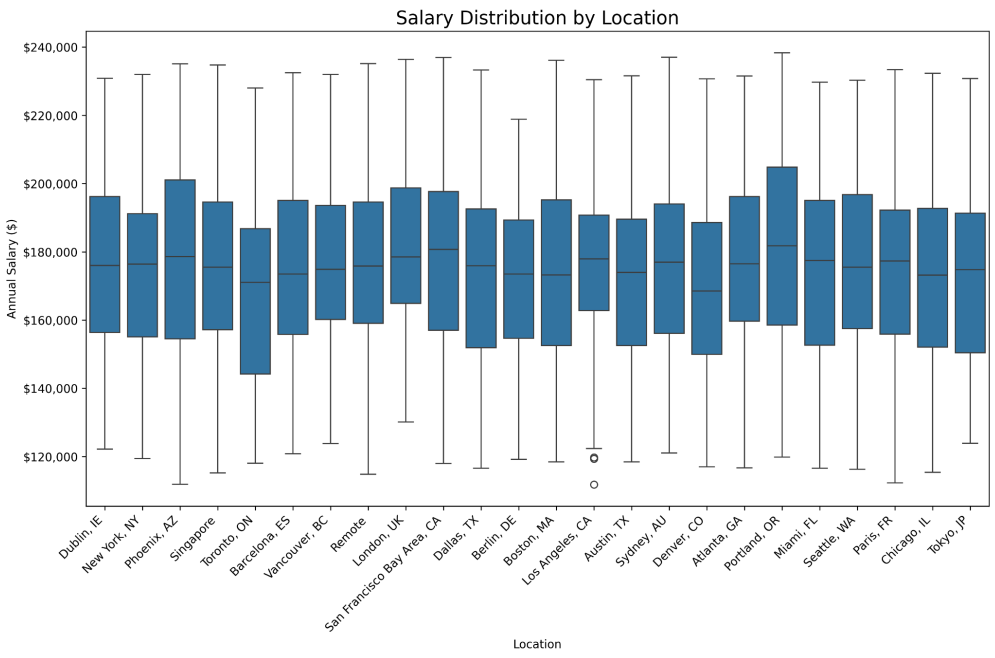
7.1 Salary Distribution by Job Title



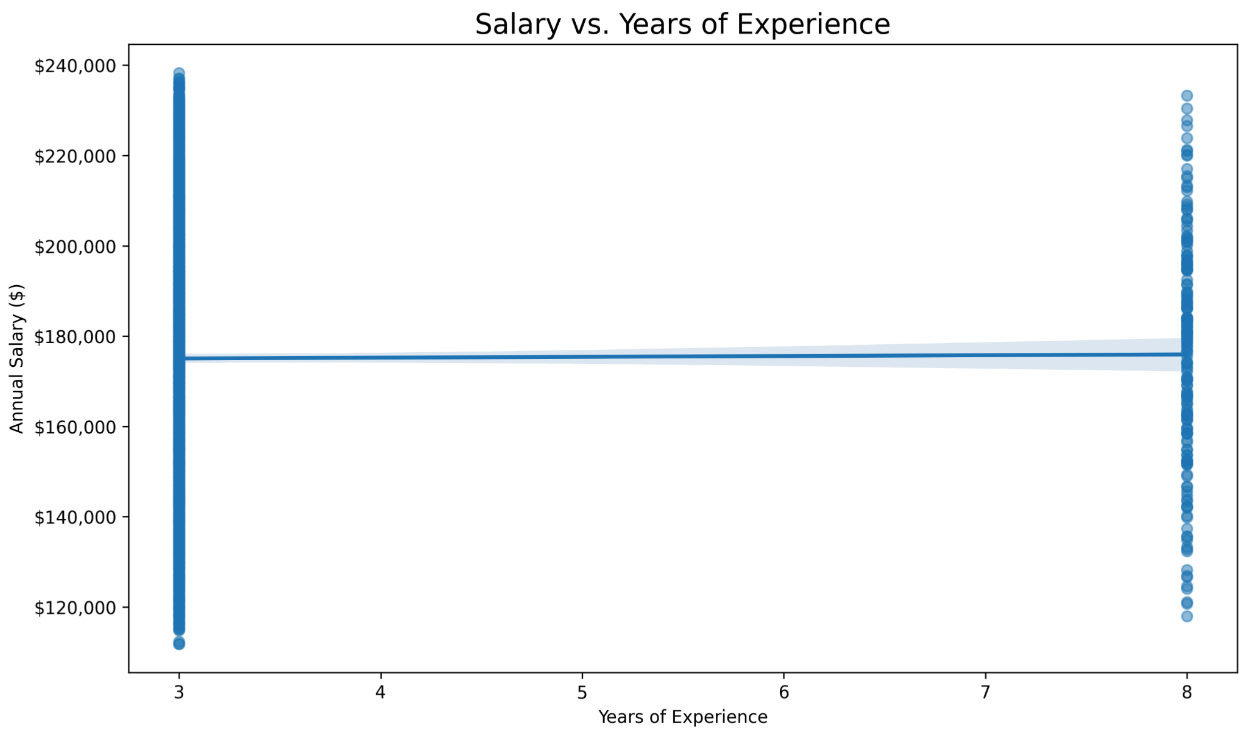
●

7.2 Salary Distribution by Location

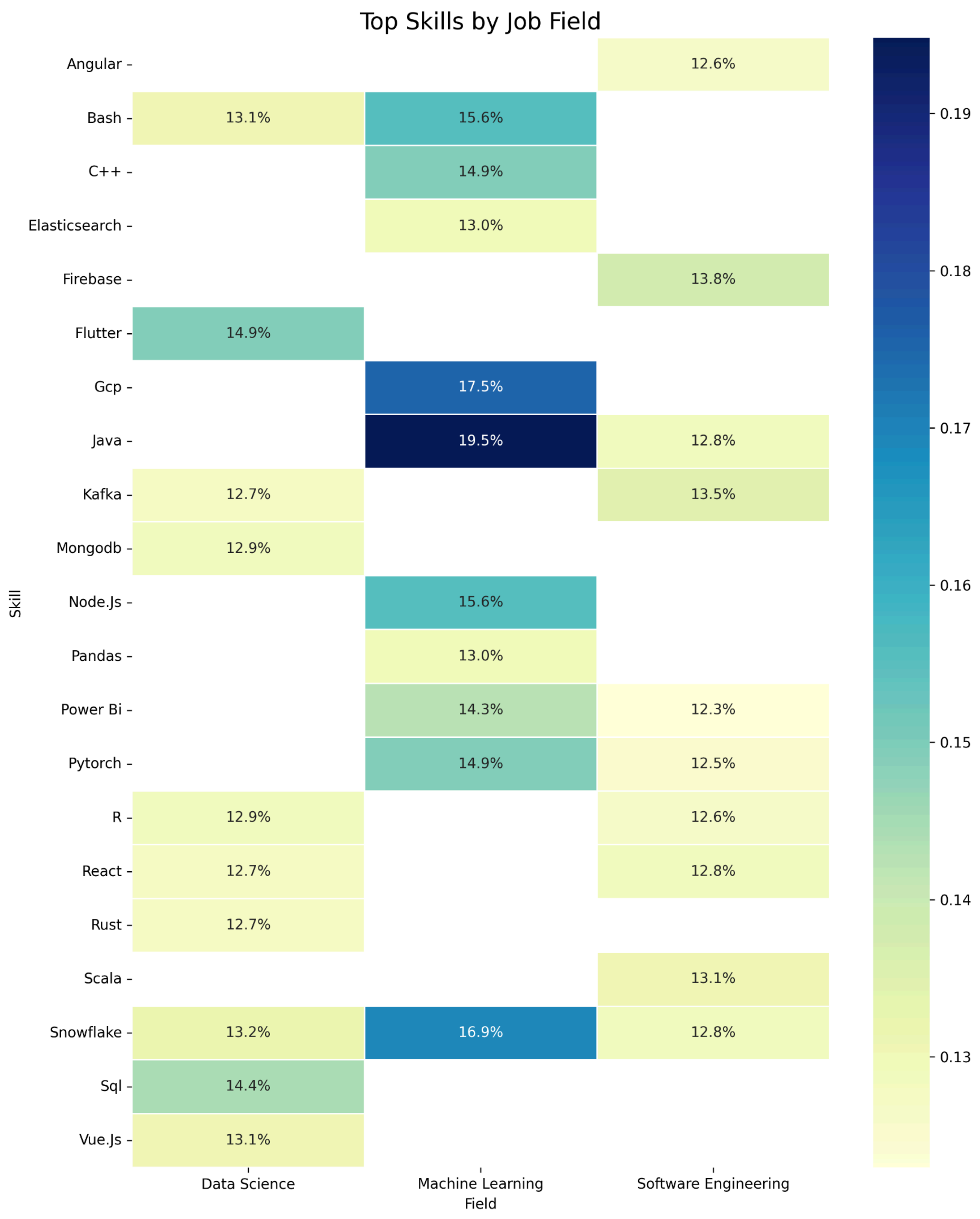
Box plots comparing salary distributions across different locations:



7.3 Salary vs. Experience



7.4 Skill Frequency Heatmap



•

8. Discussion and Conclusions

8.1 Project Findings

1. **Model Performance Challenges**:
    - Both models showed poor predictive performance with negative R^2 scores, indicating they performed worse than simply predicting the mean salary
    - The Random Forest model ($R^2 = -0.050$) and Gradient Boosting model ($R^2 = -0.022$) both struggled to capture the complex relationships in the data
    - High MAE values (over $21,000) suggest substantial prediction errors relative to typical tech salaries
2. **Feature Importance Insights**:
    - The total number of skills required ("skills_count") was consistently the most important feature in both models
    - Company size appears to influence salary predictions significantly
    - Certain job titles (Data Analyst, Software Engineer, Full Stack Engineer) have higher importance than others
    - Specific technical skills like SQL, Kafka, React, and Firebase show notable importance
    - Kubernetes and Node.js were particularly important in the Gradient Boosting model
3. **Field Analysis**:
    - The dataset contained significant numbers of records for Software Engineering (919 jobs) and Data Science (597 jobs)
    - This suggests these fields are well-represented in the job market and have sufficient data for analysis
4. **Potential Data Limitations**:
    - The negative R^2 scores suggest potential issues with data quality, feature relevance, or model assumptions
    - Salary information may be inconsistent or limited in the collected data
    - Geographic salary variations may not be adequately captured by the features

8.2 Challenges Encountered

1. **Data Collection Challenges**:
    - Many job postings didn't include salary information, reducing the training dataset size
    - Website structure changes required frequent scraper updates
    - Anti-scraping measures necessitated delays and rotating user agents
    - Some sites blocked automated access entirely
2. **Data Quality Issues**:
    - Inconsistent salary formats (hourly, annual, ranges)

    - Non-standardized job titles across companies
    - Vague or missing experience requirements
    - Inconsistent skill terminology across postings
3. **Modeling Challenges**:
    - Small dataset size limited model complexity
    - High dimensionality after one-hot encoding categorical features
    - Risk of overfitting due to numerous features relative to sample size
    - Difficulty capturing regional cost-of-living differences

8.3 Recommendations for Improvement

1. **Data Collection Enhancements**:
    - Focus on collecting more job postings with explicit salary information
    - Implement more consistent salary extraction techniques
    - Target specific regions to increase geographic representation
    - Consider using paid APIs from job platforms to access structured salary data
    - Collect additional company information to provide context for salary figures
2. **Feature Engineering Improvements**:
    - Develop better normalization techniques for salary data to account for regional differences
    - Create composite features that might better represent job value (e.g., combining experience and skills)
    - Apply dimensionality reduction techniques to handle the large number of categorical features
    - Incorporate external data sources for cost-of-living adjustments
    - Use NLP techniques to extract more nuanced skill information from job descriptions
3. **Model Refinements**:
    - Try simpler models first (like Linear Regression) to establish a baseline
    - Implement proper hyperparameter tuning using grid search or random search
    - Apply regularization techniques to prevent overfitting
    - Consider specialized models for different job categories or regions
    - Use k-fold cross-validation to get more reliable performance estimates
    - Try models that handle categorical data better natively (like CatBoost)
4. **Statistical Analysis**:
    - Perform exploratory data analysis to identify potential issues with the data
    - Check for multicollinearity among features
    - Look for outliers in salary data that might be skewing predictions
    - Analyze the distribution of salaries to see if transformations are needed

- Consider log-transforming the salary data if it's right-skewed