Machine Learning Project: Salary Prediction for Job Postings

Author: Samuel Vierny
Date: 01/12/2024

Kaggle competition 1: Salary Prediction for Job Postings

- **Type of ML Task:** Regression
    - Predicting numerical values (salaries) based on job posting data.
- **Dataset:**
    - Contains job posting metadata like job descriptions, titles, and possibly categorical features like location or industry.
    - Preprocessing involves handling text and numerical data, encoding categorical variables, and managing missing values.
- **Benefits:**
    - Straightforward regression task for beginners, focusing on common preprocessing techniques like handling text and categorical data.
    - Practical real-world application, making results interpretable and engaging.
    - Lower computational overhead compared to tasks with unstructured or large datasets (e.g., images or audio).
- **Limitations:**
    - Text data requires careful preprocessing, which might introduce complexity (e.g., tokenization, embeddings).
    - Possible skewed salary distribution could affect model performance.

Kaggle competition 2: Music Hackathon

- **Type of ML Task:** Regression or Classification
    - Could involve predicting numerical music properties or classifying genres or user preferences.
- **Dataset:**
    - Includes audio features (e.g., tempo, loudness) and metadata.
    - Requires preprocessing of audio data using techniques like spectral analysis or feature extraction.
- **Benefits:**
    - Engaging and creative domain, appealing for music enthusiasts.
    - Opportunity to work with Passion subject.

- **Limitations:**
    - High computational requirements, especially for audio file preprocessing.
    - Audio data preprocessing requires specialized knowledge of signal processing.
    - Not easy for first project

Kaggle competition 3: Classifying the Brain on Music

- **Type of ML Task:** Classification
    - Predict brain activity patterns based on music-related stimuli.
- **Dataset:**
    - Includes EEG or fMRI data, which are high-dimensional time-series signals.
    - Preprocessing involves noise filtering, feature extraction, and dimensionality reduction.
- **Benefits:**
    - Unique interdisciplinary challenge combining neuroscience and machine learning, very exciting!
    - Opportunity to apply techniques for time-series or high-dimensional data.
- **Limitations:**
    - Very complex for a first project, requiring more expertise in potentially neuroscience and ML.
    - Computationally expensive and prone to overfitting due to the high-dimensional nature of the data.

Why Kaggle competition 1 - Salary Prediction for Job Postings is the my choice

- **Alignment with Experience Level:**
    - Kaggle competition 1 has manageable complexity for a first task, with data that's easy to understand and preprocess.
- **Focus on Core ML Skills:**
    - It allowed me to focus on supervised learning fundamentals, such as feature engineering, model tuning, and evaluation, without being overwhelmed by domain-specific complexities.
- **Balanced Challenge:**
    - Preprocessing job descriptions (text data) introduced a challenge without being overly complex. The potential for a fantastic learning opportunity.
- **Interpretability of Results:**

    - Predicting salaries is a relatable and interpretable task, which helps in understanding model performance and potential biases.

- **Data set characteristics:**
    - The dataset comprises structured data with a mix of numerical and categorical features. This provides a balanced challenge that is appropriate for a first project, allowing me to practice preprocessing techniques like encoding categorical variables and handling missing values.
    - Exposure to Real-World Data Issues: Working with job postings introduces common data issues such as high cardinality in categorical features and unstructured text in job descriptions. This gives me practical experience in cleaning and preparing data.
    -

- **Computational Requirements:**
    - The dataset is structured and less computationally demanding compared to audio or neuroscience datasets. It is more feasible for faster experimentation and result findings.

- **Opportunity for Feature Engineering:**
    - The inclusion of job descriptions and skills presents an opportunity to explore flattening. A process I had to do for multivalued columns I wanted to perform one hot encoding on, such as Skills in the project.
    - Features like 'Company' and 'Job Title' have many unique values, allowing you to experiment with encoding methods like frequency encoding or target encoding. This helps in understanding how to handle variables that could otherwise introduce sparsity into the dataset.

- **Relevance and significance:**
    - Salary prediction is a topic with a real-world importance. Accurate models can benefit job seekers, employers, and policy makers by providing insights into salary trends based on various job attributes.

- **Not passion subject but good beginning:**
    - While Music related assessments might be more related to my passions and interests, Kaggle competition 1 still realistically the most relevant and fitting for my assessment. I will endeavour to take on Kaggle competition 2 and 3 another time or in the future once I have strengthened my machine learning skills.

Competition Selection

I chose the "Salary Prediction for Job Postings" competition because it aligned well with my experience level and allowed me to focus on core machine learning skills. This competition offered a practical application and the chance to dive deep into data preprocessing and feature engineering.

Kaggle competition 1: Salary Prediction for Job Postings

Data Description

The dataset included job postings enriched with various features:

- Numerical Features: Metrics like company scores, director scores, and reviews.
- Categorical Features: Attributes such as job titles, company names, locations, sectors, and skills.
- Text Data: Job descriptions and skills, often presented in multivalued formats.
- Target Variable: Mean_Salary, representing the average salary for each job posting.

The data had both single-valued and multivalued columns, with some features showing high cardinality due to a large number of unique values.

Model Development

Overview

I followed a structured approach with the following stages:

1. **Exploratory Data Analysis (EDA) (performed in old code file)**
2. **Data Cleaning and Preprocessing**
3. **Feature Engineering (performed in old code file, overall performed worse)**
4. **Model Selection and Hyperparameter Tuning**
5. **Model Evaluation and Refinement**

---

1. Exploratory Data Analysis (EDA) (performed in old code file)

I knew that understanding the data thoroughly was crucial. During the EDA phase, I:

- **Analyzed Data Structure:** Identified numerical and categorical features. I paid special attention to multivalued columns like Skills, Job, and Company.
- **Assessed Missing Values:** Evaluated the extent and distribution of missing data across the dataset.
- **Examined the Target Variable:** I analyzed the distribution of Mean_Salary to detect any skewness or outliers that could affect the model.

---

2. Data Cleaning and Preprocessing

Handling Missing Values:

- **Numerical Columns:** I imputed missing values using the median since using the mean gave me worse results.
- **Categorical Columns:** Replaced missing entries with the placeholder "unknown" to retain a signal missingness overview.

Scaling Numerical Features:

- I applied the **StandardScaler** to standardize numerical features. Using MinMaxScaler didn't work as well for me, as it gave worse results.

---

3. Feature Selection (performed in old code file, overall performed worse)

Detailed Feature Importance Analysis:

- The CatBoostRegressor model revealed that 'Jobs_Group' had the highest feature importance at 22.76%, followed by 'Profile' at 8.92%, and 'State' at 6.49%. These features significantly influenced the salary predictions, indicating the strong impact of job type and location on salary. Jobs group had too many unique values to one hot encode, along with State, but Profile was one hot encoded.

- Different methods, such as Random Forest feature importance and Recursive Feature Elimination (RFE), identified key features:

    - Both Random Forest importance and RFE consistently highlighted 'Jobs_Group', 'Profile', 'State', and 'Skills' as top predictors. This overlap reinforces the relevance of these features in salary prediction.

- Impact of Preprocessing Decisions:

    - By increasing the one-hot encoding threshold from 25 to 139, I included more categorical features like 'Sector' in the encoding process. This adjustment improved (in my old code file previously) the RMSE from $32,498.73 to $28,863.48 and increased the R^2 score from 0.4494 to 0.5655. This was a definite improvement in accuracy.

Encoding Categorical Variables:

- **Low-Cardinality Features:** I implemented **One-Hot Encoding (OHE)** for features with 139 or fewer unique values (like State and Sector) to capture categorical distinctions without creating too many features.
- **High-Cardinality Features:** For features with a high number of unique values (like Skills and Company), I used **frequency encoding** in my final pipeline. In my Jupyter notebook, I had tried **target encoding** instead.

Processing Multivalued Columns:

- **Skills:** I flattened multivalued entries to extract individual skills and then used one hot binary encoding to represent the presence of each skill.

- **Other Multivalued Features:** Applied similar parsing and encoding techniques to Job, Company, and Director to accurately capture their contributions.

---

4. Model Selection and Hyperparameter Tuning

Initial Modeling with PyCaret in Jupyter Notebook:

- I started in a Jupyter notebook, using **PyCaret** for quick prototyping and comparing various algorithms.
- **GradientBoostingRegressor** performed best here in the end, although catboost did previously.
- We selected CatBoostRegressor as our final model because it outperformed other models like GradientBoostingRegressor in both RMSE and R^2 metrics.
- **Challenges Encountered:** The high dimensionality from OHE led to memory constraints, so I needed to find alternative approaches. This is when I retrospectively set the threshold of 139, and didn't also apply it to the features with many more unique values.

Transition to a Custom Pipeline:

- Realizing it was tricky to preprocess the test data like the training data, I built a custom preprocessing and modeling pipeline using scikit-learn and CatBoost.
- **Preprocessing Pipeline:** I created separate pipelines for preprocessing, training, and prediction to ensure reusability.
- **Hyperparameter Tuning:** I optimised for **CatBoostRegressor** and tuned parameters like depth, learning_rate, and iterations.

---

5. Model Evaluation and Refinement

Performance Metrics:

- **MAPE:** Served as the primary evaluation metric, as stated in the competition description.
- **MAE and R^2:** Provided additional insights into the model's predictive accuracy and explanatory power.

Model Refinements:

- **Dimensionality Reduction:** AI had to address high dimensionality by adjusting OHE thresholds and applying feature selection methods to reduce computational load without sacrificing predictive capability.
- **Feature Importance Analysis:** I had to focus on the features contributing significantly to cumulative importance, streamlining the model and enhancing interpretability.

Final Model:

- Trained a **CatBoostRegressor** on the refined feature set with optimized hyperparameters.
- **Results:** CatBoost performed best here, achieving improvements in MAPE, RMSE and R^2 compared to initial models.

Reflections on the Modeling Process

Challenges Faced

1. **Two Code File Journeys and Their Differences:**
    - **Final Code Pipeline:**
        - I frequency encoded features and set a threshold of 139 unique values for OHE.
        - Used **StandardScaler** because MinMaxScaler gave me worse results.
        - **CatBoostRegressor** performed best in this setup.
    - **Jupyter Notebook:**
        - I used **target encoding** for features above the threshold.
        - I applied **MinMaxScaler** to have values similar to the 0 and 1 from OHE. But this performed worse
        - **GradientBoostingRegressor** performed best here.
2. **Swapping Between Pipeline and Jupyter Notebook:**
    - **Issue:** I spent a lot of time switching between building a complex pipeline and working in the Jupyter notebook to encode and scale the data with more finesse, and even considered removing outliers.
    - **Outcome:** Despite the additional effort, the model from the notebook performed worse than my original pipeline model, and I'm still not sure why.
3. **Handling High Dimensionality:**
    - **Issue:** Applying OHE to high-cardinality features like Job and Company resulted in too many features, leading to computational inefficiency.
    - **Solution:** I split OHE at 139 unique values, which was a good decision. For features above that threshold, I used frequency encoding in the pipeline and had tried target encoding in the notebook. I suspected target encoding would have performed better, but it didn't in the overall set up.
4. **Pipeline Development:**
    - **Issue:** The initial code structure in jupyter lacked reusability and consistency for the training and test datasets.
    - **Solution:** I decided to make a preprocessing pipeline to standardize data transformation across all datasets.

Key Learnings

- **Stick to One Way:** In the future, I'll stick to one method—pipeline or notebook—to avoid unnecessary complexity and double work.
- **More EDA:** I'll do more exploratory data analysis and look at the data shape, columns, and values after each step. It's worth not going back and changing a bunch of stuff, and instead making intelligent decisions as I go.
- **Follow a Step-by-Step Procedure:** I'll follow a structured procedure more closely and avoid trying to see the end result too soon. Understanding each part leads to a better final model.
- **Importance of Reproducibility:** Developing modular and well-documented code means I can reuse it and refine anything again a lot easier without abstraction and confusion.

Personal Reflection

This project was challenging, and I wish I had more time to complete it or try another dataset. I think I should have gone with a different competition on Kaggle, although it is tricky to foresee that before having ever done one. The code was tough to implement, but I'm proud of how much I learned. In every area, coding, machine learning strategies, and organization, I have learnt a tremendous amount throughout this Kaggle competition. I have probably spent close to 30 hours on it in total, which is far too much for the output. I have used an abundance of libraries I have never heard of, along with creative techniques and refinements, but it is unfortunate I could not score higher, and potentially with each new creative solution I had, the project got more complex and something must have gone wrong in the code (especially the old code file)

I have realized that it's tempting to be quick and just see how the model performs, but taking the time to understand each part is crucial. If I understand each step, I can make intelligent decisions and refine the model effectively. Like in other areas and industries too, if you go to far and have to change something way further back in the pipeline, then you need to consider changing every that comes after it again too! This is a big learning lesson throughout this project.

Conclusion

The "Salary Prediction for Job Postings" competition gave me a practical application of supervised learning techniques to a real-world problem. Through systematic exploration, meticulous preprocessing, and iterative modelling, I have managed to develop many attempts at predictive model that estimates mean salaries with reasonable accuracy. The challenges I encountered reinforced the importance of adaptability, thoroughness, and continuous learning in the field of machine learning.

Generative AI was used for helping structure this report based on my findings in my code. The views and reflections expressed are my own.

Future Outlook

In future projects, I'll:

- Stick to one approach to avoid unnecessary complexity.
- Conduct more thorough EDA to make informed decisions at each step.
- Follow a step-by-step procedure to build a solid foundation before evaluating the model.
- Understand that patience and attention to detail lead to better outcomes.

I'm excited to apply these learnings to my next machine learning challenge. I definitely had a fun time, and wish I had more time to refine my strategies for this assessment.

Kaggle Competition Details

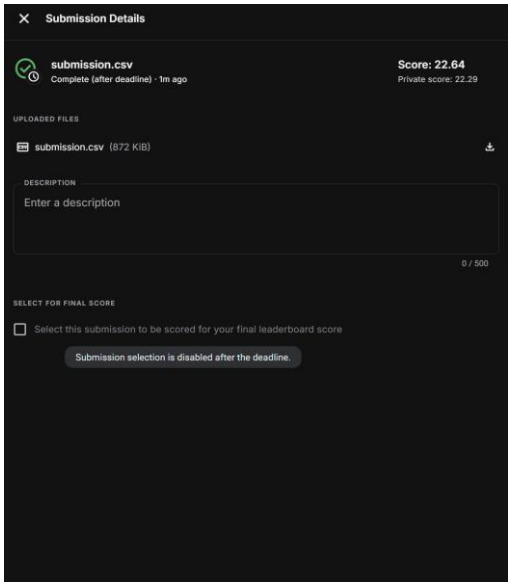
- **Kaggle Username**: *Samuel Vierny*
- **Competition Name**: Salary Prediction for Job Postings
- **Number of Teams**: *83*
- **Leaderboard Position**: *60 (if could submit it)*

Please refer to the attached screenshot for verification of the leaderboard position.

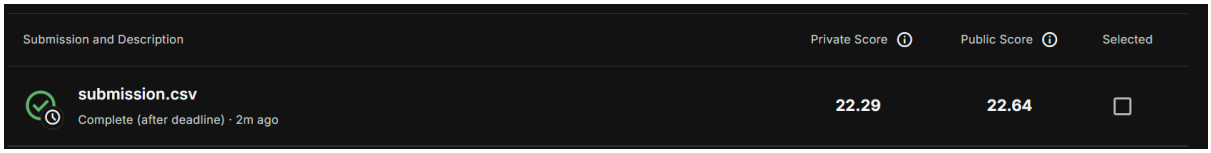
Attachments

- **Leaderboard Screenshot**

I could not access the leaderboard, because the competition somehow would not let me as it was disabled... See the following screenshot.



However below is my score (one of the many) that I managed with the pipeline code that is accessible for you too in the attachments.



Unfortunately, I would have place 60 in the leaderboard, out of then 84. This is a shame I and I would have liked to have done better.

- **Model Files and Code**: Submitted as a zip file containing the preprocessing pipeline, training script, and prediction script.
