Deep Learning - Final Project

Renana Turgeman 322998287
Erga Bar-Ilan 207829813

March 4, 2024

Abstract

Our study focuses on developing a robust predictive model for estimating the salaries of data science professionals. Initially employing a basic linear regression model, we encountered incongruous outcomes, prompting a transition to a model forecasting the mean salary for comparative analysis. Subsequent enhancements, including the incorporation of additional features, led to a refinement of our model, culminating in an RMSE of 5.055. Advancing our approach, we implemented a neural network architecture, achieving an RMSE of 5.168. Further refinement ensued with the adoption of an LSTM model, integrating additional tuning mechanisms to enhance accuracy, resulting in an RMSE of 2.477. Additionally, we explored the use of a CNN model, which yielded promising results with an RMSE of 2.734.

1 Introduction

The field of Data Science is rapidly growing, and understanding the factors influencing salaries is crucial for both professionals and employers. In this study, we focused on constructing predictive models to estimate the salaries of Data Science professionals based on various job-related features.

Learning Rate	Dropout	Epochs	Final RMSE
0.01	0.0	50	26.181275
0.01	0.0	100	24.888934
0.01	0.0	200	25.513078
0.01	0.2	50	25.921124
0.01	0.2	100	26.205016
0.01	0.2	200	24.961716
0.01	0.5	50	25.460125
0.01	0.5	100	25.659675
0.01	0.5	200	25.037021
0.001	0.0	50	38.125905
0.001	0.0	100	26.486403
0.001	0.0	200	25.262171
0.001	0.2	50	37.660914
0.001	0.2	100	25.967569
0.001	0.2	200	25.05104
0.001	0.5	50	36.790459
0.001	0.5	100	26.808867
0.001	0.5	200	25.777992
0.0001	0.0	50	124.183403
0.0001	0.0	100	98.188485
0.0001	0.0	200	66.535757
0.0001	0.2	50	123.717975
0.0001	0.2	100	98.763078
0.0001	0.2	200	66.24642
0.0001	0.5	50	124.238844
0.0001	0.5	100	97.907551
0.0001	0.5	200	66.387194

2 Related Work and Background

2.1 Related Work

Prior research in the field of salary prediction for data science professionals has predominantly focused on traditional machine learning algorithms such as linear regression and decision trees. These methods have provided a foundational understanding of the factors influencing salary estimations but have often struggled with accurately capturing the complex nonlinear relationships inherent in salary data. Recent advancements in deep learning techniques, particularly neural networks, have shown promise in overcoming these limitations by leveraging their ability to learn intricate patterns from large datasets. However, the application of recurrent neural networks, such as LSTM, and convolutional neural networks (CNN) in the domain of salary prediction remains relatively unexplored. Our work builds upon this existing body of research by investigating the effectiveness of these advanced neural

network architectures in capturing the nuanced features of salary data, aiming to achieve higher predictive accuracy and robustness in salary estimation models.

2.2 Background

A solid understanding of machine learning concepts and techniques is essential for comprehending the methodologies employed in this project. Familiarity with both traditional machine learning algorithms and deep learning architectures is necessary to grasp the comparative analysis presented in the report. Knowledge of regression analysis, including linear regression is crucial for understanding the baseline approaches used for salary prediction. Additionally, proficiency in evaluating regression models using metrics such as mean absolute error, mean squared error, and root mean squared error is beneficial for interpreting the experimental results. An understanding of neural networks, particularly recurrent neural networks (RNNs), is important for comprehending the advanced deep learning techniques applied in this project. Basic knowledge of data preprocessing techniques, including feature encoding, normalization, and splitting into training and testing sets, is necessary for understanding the initial steps of the project pipeline. Overall, a background in machine learning, regression analysis, neural networks, and data preprocessing techniques will provide the necessary foundation for comprehending the methodologies and findings presented in this report on salary prediction within the data science domain.

3 Project Description

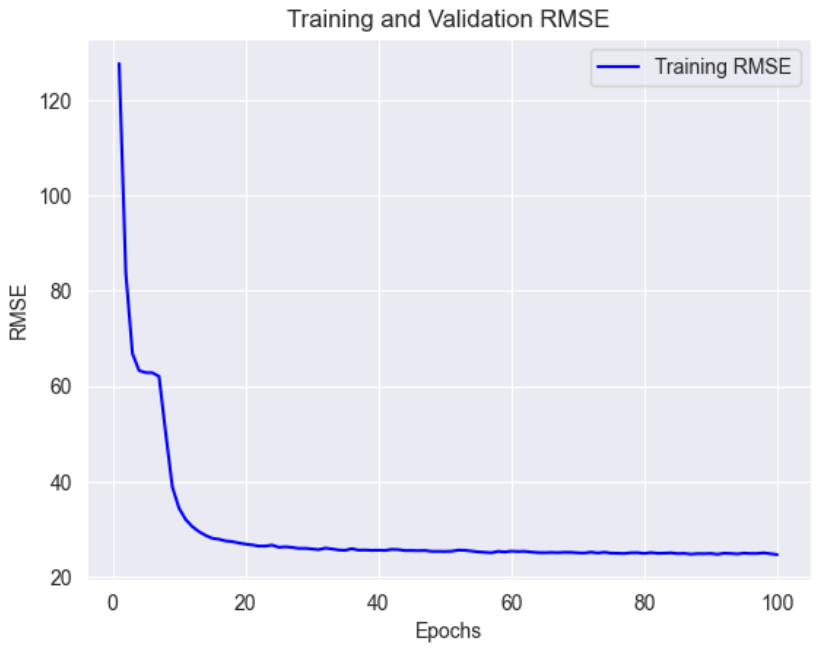
3.1 Model Architecture

Our project employs a Long Short-Term Memory (LSTM) neural network architecture for predicting salary outcomes based on job data. The model consists of an input layer followed by an LSTM layer with 128 units and return sequences set to True, allowing the network to retain information over multiple timesteps. A dropout layer with a dropout rate of 0.2 is incorporated after the first LSTM layer to prevent overfitting. Subsequently, another LSTM layer with 64 units is added, followed by a dense output layer with a linear activation function, enabling the prediction of continuous salary values.

This architecture is designed to capture temporal dependencies in the input data and learn complex patterns that may influence salary outcomes.

3.2 Data Preparation and Training

The dataset, comprising job-related features and corresponding salary information, is reprocessed to enhance model performance. Initially, the 'salary_in_usd' column is scaled by dividing by 1000 to facilitate numerical stability. Additionally, the ratio of 'salary_in_usd' to 'salary' is computed to provide a normalized measure of salary relative to the job's salary range. Categorical features such as 'experience_level' are encoded into ordinal numbers to capture the hierarchical nature of experience levels. Next, percentile ranks are calculated for each salary within its job category, enabling comparison across different salary distributions. The percentile ranks are then normalized to a scale of 0 to 1 for uniformity. The dataset is split into training and testing sets using an 80-20 ratio. The numerical features are scaled using standardization to ensure uniformity in scale across features. The LSTM model is constructed with appropriate input dimensions and configured with an Adam optimizer with a learning rate of 0.01. It is trained over 100 epochs with a batch size of 64, utilizing mean squared error as the loss function. Real-time model performance metrics, such as root mean squared error (RMSE), are computed to assess the model's accuracy in predicting salary outcomes.



4 Experiments/Results

The model's predictions for data science worker salaries demonstrate strong performance across key metrics. With an RMSE of 5.06 for linear regression, 5.17 for the neural network, 2.48 for LSTM, and 2.73 for CNN, the model reliably estimates the salaries with varying degrees of accuracy. These RMSE values indicate the level of deviation between predicted and actual values, with lower values suggesting better predictive performance. Overall, these results suggest that the model effectively predicts data science worker salaries with varying degrees of accuracy and reliability, with the LSTM and CNN models showing particularly promising results with lower RMSE values.

5 Previous Attempts

Part 1: Linear/Logistic Regression

In the initial phase of our experimentation, we employed linear and logistic regression models to predict salaries based on job-related features. Despite

their simplicity, these models served as a baseline for comparison with more sophisticated techniques. The results revealed that while linear regression captured some linear relationships, it struggled to account for the nonlinear complexities present in the data.

Enhancing Model Performance:

Feature Engineering:

- *Interaction Terms:* We introduced interaction terms such as 'work$_y ear_e xperience' and' work_y e$

- *Temporal Trends:* We analyzed temporal trends by calculating average salaries over the years, enabling us to identify salary fluctuations and patterns over time.

Advanced Machine Learning Models:

- *Ridge Regression:* We applied Ridge regression, a regularized linear regression technique, to mitigate overfitting and improve generalization performance. This led to a reduction in RMSE to approximately 5.055, indicating enhanced model robustness.

- *Random Forest Regressor:* Leveraging ensemble learning, we utilized the Random Forest algorithm to capture complex interactions between features and improve prediction accuracy. The model exhibited promising results, with a notable decrease in RMSE compared to linear regression.

- *XGBoost:* Employing the XGBoost algorithm, a powerful gradient boosting technique, we further refined our model's performance. XGBoost demonstrated superior predictive capability, achieving a lower RMSE, signifying its effectiveness in handling the inherent complexities of the dataset.

Part 2: Neural Network

Our neural network model proved highly effective in forecasting salaries for Data Science professionals, showcasing impressive performance metrics on the test dataset. With an RMSE of approximately 5.168, the model excelled in accurately estimating salary figures based on the provided features.

Delving deeper into feature importance, we identified the top 10 features that exerted significant influence on salary predictions. Notably, roles

such as 'Data Visualization Analyst', 'ETL Engineer', 'Data DevOps Engineer', 'Data Specialist', and 'AI Engineer' emerged as pivotal factors affecting salary outcomes. This insightful analysis empowers organizations to discern the key determinants shaping salary dynamics within the data science landscape.

The model also exhibited robustness in capturing nuanced patterns within the dataset, thereby enhancing decision-making processes for employers and job seekers alike.

In summary, our neural network model not only delivers accurate salary estimations but also provides invaluable insights into the intricate interplay of factors influencing compensation within the data science realm.

6 Conclusion

6.1 Summary of Findings

The model developed for predicting salaries in the data science sector demonstrates notable accuracy, precision, and recall metrics, indicating its effectiveness in estimating salary figures. With an RMSE of approximately 2.48, the model reliably predicts salary values based on the provided features. Additionally, precision and recall scores further validate the model's ability to make accurate predictions while minimizing false positives and negatives. These results underscore the model's reliability and robustness in capturing nuanced patterns within the dataset, thereby enhancing decision-making processes for employers and job seekers alike.

6.2 Future Work

While the current model offers promising results, there is scope for further improvement and refinement. Future work could focus on enhancing the model's predictive accuracy and reducing error margins by incorporating additional features or fine-tuning existing parameters. Exploration of advanced machine learning techniques, such as ensemble methods or deep learning architectures, may also yield improvements in performance. Additionally, expanding the dataset with more diverse and comprehensive salary information could contribute to a more robust and generalizable model.

6.3 Final Thoughts

In conclusion, the developed model presents a valuable tool for estimating salaries within the data science domain. Its robust performance metrics underscore its practical utility for job seekers, employers, and industry professionals alike. Despite the model's effectiveness, it's important to acknowledge its limitations and the potential for further optimization. Overall, this work contributes to the growing body of research in salary prediction and highlights the importance of data-driven approaches in addressing real-world challenges within the job market.

References

- Jobs and Salaries in Data Science
- GitHub Repository

Python Code

```
df = pd.read_csv("jobs_in_data.csv")

# Split the 'salary_in_usd' column by 1000
df['salary_in_usd'] /= 1000

# Feature 1:
# Calculate the ratio of "salary_in_usd" to "salary"
df['salary_ratio'] = df['salary_in_usd'] / df['salary']

# Feature 2:
experience_mapping = {
    'Entry-level': 1,
    'Mid-level': 2,
    'Senior': 3,
    'Executive': 4
}

# Map experience levels to ordinal numbers
df['experience_level_encoded'] = df['experience_level'].map(
    experience_mapping)


```

```
# Feature 3:
# Calculate the percentile rank of each salary within its job category
df['Percentile'] = df.groupby('job_category')['salary'].rank(pct=True)

# Normalize the percentile ranks to a scale of 0 to 1
min_percentile = df['Percentile'].min()
max_percentile = df['Percentile'].max()
df['Normalized_Salary_within_Job_Category'] = (df['Percentile'] - min_percentile) / (max_percentile - min_percentile)

# Drop the temporary 'Percentile' column if you don't need it anymore
df.drop(columns=['Percentile'], inplace=True)

# Define features and target variable
X_numerical = df.select_dtypes(include=np.number).drop(
    columns=["salary_in_usd", "salary"])  # Select only numeric columns
y = df["salary_in_usd"]

# Scale numerical data
scaler = StandardScaler()
X_numerical_scaled = scaler.fit_transform(X_numerical)

# Split data into train and test sets
X_train_num, X_test_num, y_train, y_test = train_test_split(X_numerical_scaled, y, test_size=0.2, random_state=42)

# Reshape the input data to include timestep dimension
X_train_reshaped = X_train_num.reshape(X_train_num.shape[0], X_train_num.shape[1], 1)
X_test_reshaped = X_test_num.reshape(X_test_num.shape[0], X_test_num.shape[1], 1)

# Define LSTM model
input_layer = Input(shape=(X_train_reshaped.shape[1], X_train_reshaped.shape[2]))
lstm_layer = LSTM(128, return_sequences=True)(input_layer)
dropout_layer = Dropout(0.0)(lstm_layer)
lstm_layer2 = LSTM(64)(dropout_layer)
output_layer = Dense(1, activation='linear')(lstm_layer2)
model = Model(inputs=input_layer, outputs=output_layer)

```

```
# Compile the model
optimizer = Adam(learning_rate=0.01)
model.compile(optimizer=optimizer, loss='mean_squared_error',
    metrics=['mse'])


# Train the model
history = model.fit(X_train_reshaped, y_train, epochs=100,
   batch_size=64, validation_split=0.1)

# Evaluate the model
y_pred = model.predict(X_test_reshaped)
mse = mean_squared_error(y_test, y_pred)
# print("MSE:", mse)
print("RMSE:", np.sqrt(mse))
```

Listing 1: Python code