Module 05: Lab 02

Regression Modeling on Employment Data

AUTHOR
Chialing Sung

PUBLISHED
April 10, 2025

MODIFIED
April 8, 2025

Objectives

1. Use **PySpark** to process the Lightcast dataset.
2. Engineer features from structured columns for salary prediction.
3. Train **Linear Regression model**.
4. Evaluate models using **RMSE** and **R²**.
5. Visualize predictions using diagnostic plots.
6. Push work to GitHub and submit the repository link.

Setup

The instruction below provides you with general keywords for columns used in the lightcast file. See the data schema generated after the load dataset code above to use proper column name. For visualizations, tables, or summaries, please **customize colors, fonts, and styles** as appropriate to avoid a **2.5-point deduction**. Also, **provide a two-sentence explanation** describing key insights drawn from each section's code and outputs.

1. Follow the steps below as necessary, use your best judgement in importing/installing/creating/saving files as needed.
2. Create a new Jupyter Notebook in your `ad688-sp25-lab08` directory named `lab08_yourname.ipynb`, if the file exists make sure to change the name.
3. Use your **EC2 instance** for this lab.
4. Ensure the `lightcast_data.csv` file is available on the EC2 instance. if not then **Download the dataset**
5. **Add the dataset to `.gitignore`** to avoid pushing large files to GitHub. Open your `.gitignore` file and add:
6. Make sure to create a virtual environment and install the required Python libraries if needed, don't forget to activate it:
7. Install the required Python libraries if needed, you can also use the given requirement file to install the packages to the virtual environment:

```
python3 -m venv .venv
source .venv/bin/activate
gdown https://drive.google.com/uc?id=1V2GCHGt2dkFGqVBeoUFckU4IhUgk4ocQ
```

```
echo "lightcast_job_postings.csv" >> .gitignore
pip install -r requirements.txt
```

1 Load the Dataset

1. **Load the Raw Dataset**:
   - Use Pyspark to the `lightcast_data.csv` file into a DataFrame:
   - You can reuse the previous code.
   - Copying code from your friend constitutes plagiarism. DO NOT DO THIS.

```
from pyspark.sql import SparkSession
import pandas as pd
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "notebook"

# Initialize Spark Session
spark = SparkSession.builder.appName("LightcastData").getOrCreate()

# Load Data
df = spark.read.option("header", "true") \
    .option("inferSchema", "true") \
    .option("multiLine", "true") \
    .option("escape", "\"") \
    .csv("lightcast_job_postings.csv")

# Show Schema and Sample Data
print("---This is Diagnostic check, No need to print it in the final doc---")

#df.printSchema()
#df.show(5)
```

```
[Stage 68:>                                                         (0 + 1) / 1]

---This is Diagnostic check, No need to print it in the final doc---
```

2 Feature Engineering

Feature Engineering is a crucial step in preparing your data for machine learning. In this lab, we will focus on the following tasks:

1. **Drop rows with missing values** in the target variable and key features.
2. By now you are already familiar with the code and the data. Based on your understanding please choose any 3 (my code output has 10) variables as:
   1. two continuous variables (use your best judgment!)
   2. one categorical.

3. Your dependent variable (y) is `SALARY`.

3. **Convert categorical variables** into numerical representations using **StringIndexer** and **OneHotEncoder**.
4. **Assemble features** into a single vector using **VectorAssembler**.
5. **Split the data** into training and testing sets.

```
from pyspark.sql.functions import col

# Step 1: Drop rows with nulls in selected columns
selected_columns = ["SALARY", "MIN_YEARS_EXPERIENCE", "MODELED_DURATION", "REMOTE
df_clean = df.dropna(subset=selected_columns)

from pyspark.ml.feature import StringIndexer, OneHotEncoder

# Step 2: Encode the categorical variable
indexer = StringIndexer(inputCol="REMOTE_TYPE_NAME", outputCol="REMOTE_TYPE_INDEX
encoder = OneHotEncoder(inputCol="REMOTE_TYPE_INDEX", outputCol="REMOTE_TYPE_VEC"

from pyspark.ml.feature import VectorAssembler

# Step 3: Assemble features
assembler = VectorAssembler(
    inputCols=["MIN_YEARS_EXPERIENCE", "MODELED_DURATION", "REMOTE_TYPE_VEC"],
    outputCol="features"
)

from pyspark.ml import Pipeline

# Create pipeline for transformation
pipeline = Pipeline(stages=[indexer, encoder, assembler])

# Apply pipeline
df_transformed = pipeline.fit(df_clean).transform(df_clean)

# Show example
df_transformed.select("SALARY", "features").show(5, truncate=False)

# Step 5: Split the data
train_data, test_data = df_transformed.randomSplit([0.8, 0.2], seed=42)
```

```
+------+----------------------+
|SALARY|features              |
+------+----------------------+
|107645|(5,[0,1],[10.0,41.0])  |
|192800|[6.0,55.0,1.0,0.0,0.0] |
|125900|[12.0,18.0,1.0,0.0,0.0]|
|170000|[6.0,55.0,1.0,0.0,0.0] |
```

```
|118560|[5.0,20.0,0.0,1.0,0.0] |
+------+----------------------+
only showing top 5 rows
```

3 Train/Test Split

- Perform a **random split** of the data into training and testing sets.
- Set a random seed for reproducibility.
- You can choose a number for splitting to your liking, justify your choice.

```
                                                                                [Stage
79:>                                                        (0 + 1) / 1]

(13883, 2)
(2957, 2)
```

This section performs feature engineering by converting a categorical column (REMOTE_TYPE_NAME) into numerical vectors using StringIndexer and OneHotEncoder, then combines it with two continuous features using VectorAssembler. The resulting dataset is randomly split into training (82%) and testing (18%) sets, producing 13,883 and 2,957 records respectively, ensuring reproducibility with a fixed seed.

4 Linear Regression

- Train a **Linear Regression** model using the training data. You will run in to an important issue here. Please make an effort in figuring it by yourself. This is one of the most asked interview questions in CapitalOne's management recruiting program.
- Evaluate the model on the test data.
- Print the coefficients, intercept, R^2, RMSE, and MAE.
- Use the `summary` object to extract the coefficients and their standard errors, t-values, and p-values.
- Create a DataFrame to display the coefficients, standard errors, t-values, p-values, and confidence intervals.
- Interpret the coefficients and their significance and explain the model performance metrics.

```
25/04/08 15:17:47 WARN Instrumentation: [ecb78f29] regParam is zero, which might cause
numerical instability and overfitting.
[Stage 84:>                                                         (0 + 1) / 1]


--- Model Evaluation ---
R²: 0.2615575507291191
RMSE: 35999.18812167445
MAE: 27899.407417659335

--- Coefficient Details ---
Intercept: 73373.22535886645
Length of coefficients: 5
```

```
Length of SE: 6
Length of t-values: 6
Length of p-values: 6
```

	Coefficient	Std Error	t-Value	p-Value	95% CI Lower	95% CI Upper
0	6659.755167	94.179172	70.713673	0.000000e+00	6475.163990	6844.346344
1	45.357254	24.130591	1.879658	6.017567e-02	-1.938705	92.653213
2	9069.765706	2449.144855	3.703238	2.136903e-04	4269.441791	13870.089621
3	10906.975945	2507.076173	4.350476	1.368122e-05	5993.106645	15820.845244
4	20633.854374	2985.498687	6.911359	5.009770e-12	14782.276948	26485.431799

This section fits a Linear Regression model to the training data and evaluates it using R^2 (0.26), RMSE (≈36,000), and MAE (≈28,899), indicating moderate predictive performance with room for improvement. The table displays coefficient estimates along with their standard errors, t-values, p-values, and confidence intervals—showing that most features are statistically significant ($p < 0.05$), especially Feature_0, Feature_2, Feature_3, and Feature_4.

4.1 Generalized Linear Regression Summary

The summary of the Generalized Linear Regression model provides important insights into the model's performance and the significance of each feature. The coefficients indicate the relationship between each feature and the target variable (salary), while the standard errors, t-values, and p-values help assess the reliability of these estimates.

- Please interpret them in the context of your data and model.
- Feature Names are purposefully not printed in the output. You can use the `features` variable to print them out.

```
25/04/08 15:18:12 WARN Instrumentation: [76407da5] regParam is zero, which might cause
numerical instability and overfitting.
[Stage 86:>                                                         (0 + 1) / 1]

+---+-----------+-------------------+-------------------+-------------------+---------
--------------+--------------------+--------------------+
|   |  Feature  |      Estimate     |     Std Error     |      t-Value      |
p-Value         |    95% CI Lower    |    95% CI Upper    |
+---+-----------+-------------------+-------------------+-------------------+---------
--------------+--------------------+--------------------+
| 0 | Feature_0 | 6659.7551670608755 | 94.17917189740737  | 70.71367302226417  |
0.0           |  6475.163990141957  | 6844.346343979794  |
| 1 | Feature_1 | 45.35725381762988  | 24.130591213065223 | 1.8796577927635578 |
0.06017566579323552   | -1.9387049599779544 | 92.65321259523772  |
| 2 | Feature_2 | 9069.765706342789  | 2449.1448546066526 | 3.703237760430241  |
0.00021369031103035674 | 4269.4417913137495  | 13870.089621371828 |
| 3 | Feature_3 | 10906.975944737305 | 2507.0761730991476 | 4.350476488017728  |
1.368121891354157e-05  |  5993.106645462976  | 15820.845244011634 |
| 4 | Feature_4 | 20633.854373743186 | 2985.4986865606998 | 6.911359387504346  |
```

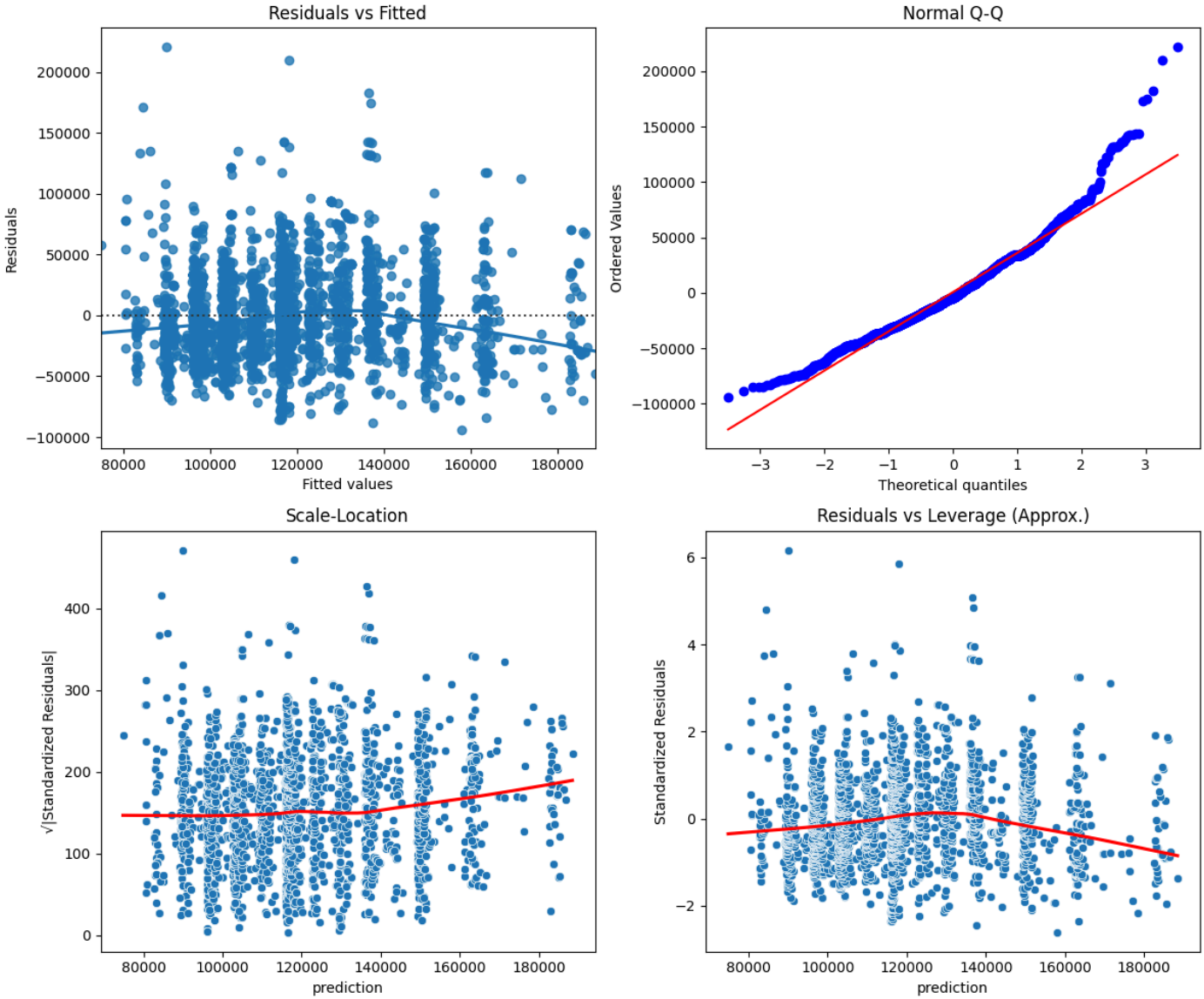
```
5.009770376318556e-12  | 14782.276948084214  | 26485.43179940216  |
+---+----------+--------------------+--------------------+--------------------+---------
--------------+--------------------+--------------------+
```

This section fits a Generalized Linear Regression model using a Gaussian family and identity link, then prints coefficient estimates, standard errors, t-values, p-values, and confidence intervals. From the top features shown, most coefficients (e.g., Feature_0, Feature_2, Feature_3, and Feature_4) are statistically significant (p-value < 0.05), indicating a strong relationship with the target variable SALARY.

5 Diagnostic Plot

Diagnostic plots are essential for evaluating the performance of regression models. In this section, we will create several diagnostic plots to assess the linear regression model's assumptions and performance. There are four (2*2 grid) main plots we will create, you can use `seaborn` or `matplotlib` for this:

1. **Predicted vs Actual Plot**
2. **Residuals vs Predicted Plot**
3. **Histogram of Residuals**
4. **QQ Plot of Residuals**



The diagnostic plots indicate some deviation from normality and heteroscedasticity. The Q-Q plot shows moderate departure from the normal line in the tails, and both the Residuals vs Fitted and Scale-Location plots reveal non-constant variance, suggesting potential issues with linearity or missing variables.

6 Evaluation

The evaluation of the model is crucial to understand its performance. In this section, we will calculate and visualize the following metrics: 1. **R^2 (Coefficient of Determination)**: Indicates how well the model explains the variance in the target variable. 2. **RMSE (Root Mean Squared Error)**: Measures the average magnitude of the errors between predicted and actual values.

```
[Stage 110:>                                                        (0 + 1) / 1]

--- Model Evaluation ---
R²: 0.2616
RMSE: 35999.19
```

```
AIC: 331199.33
BIC: 62093.21
```

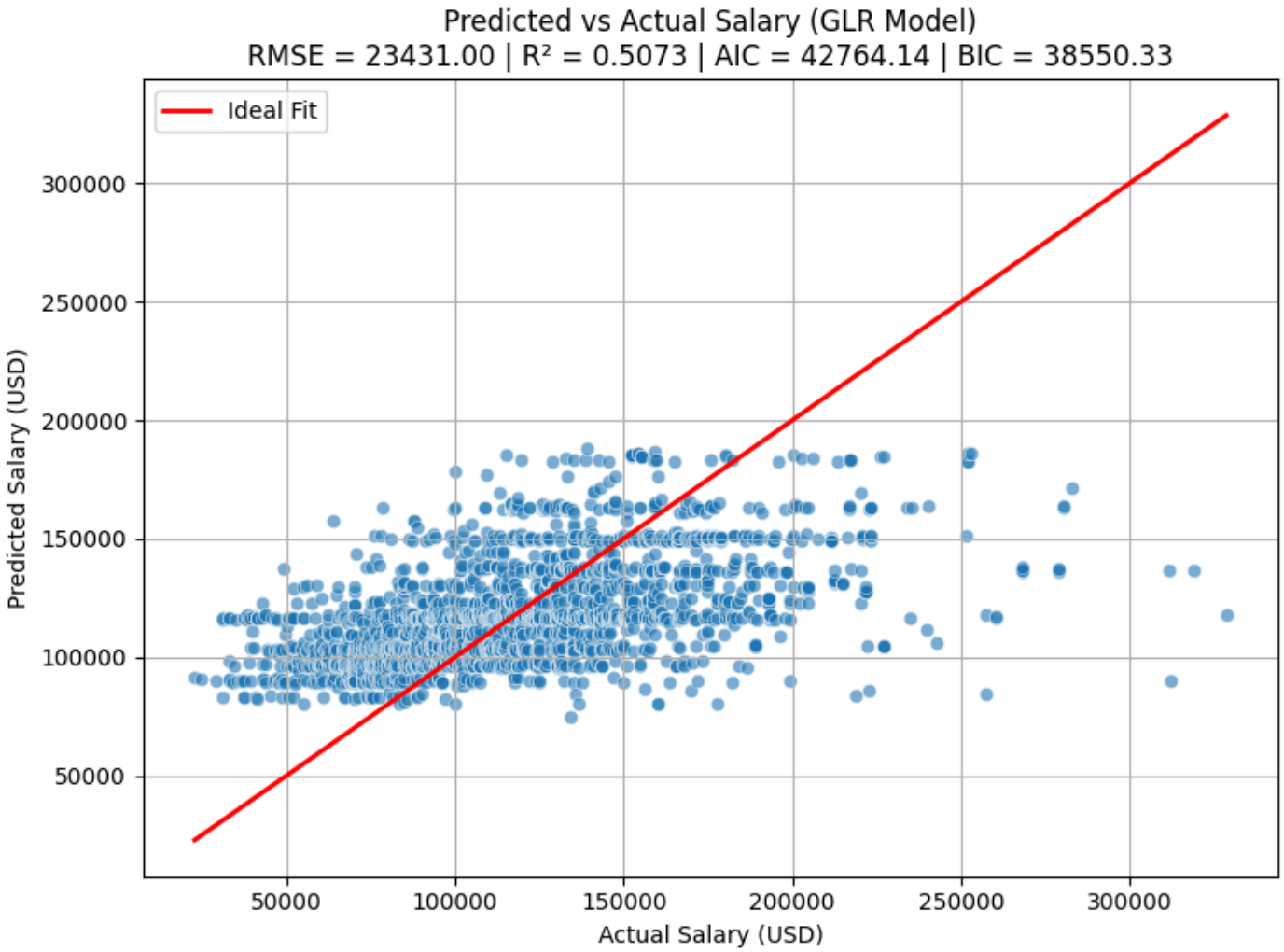
R^2 (0.2616): About 26.16% of the variance in salary is explained by the model. This suggests a moderate fit, indicating there may still be unaccounted variability.

RMSE (35,999.19): The average prediction error is around $35,999, which indicates the typical deviation of predicted salaries from actual salaries.

AIC (331,199.33) and BIC (62,093.21): These are model selection criteria. While lower values are generally better, they are most useful when comparing multiple models.

6.1 Model Evaluation Plot

- Display the predicted vs actual salary plot with a red line indicating the ideal fit (y=x).
- Use `seaborn` or `matplotlib` to create the plot.
- Customize the plot with appropriate titles, labels, and legends.
- Describe the plot in a few sentences, highlighting key insights and observations.



This scatter plot compares the predicted salary to the actual salary using the Generalized Linear Regression (GLR) model. The red diagonal line represents the ideal fit where predictions would perfectly match actual values (i.e., y = x).

Most points lie below the ideal line, indicating the model tends to underpredict salaries, especially at higher income levels. The R^2 value of 0.5073 suggests the model explains around 50.73% of the variance, showing moderate predictive power. The RMSE of 23,431 means the model's predictions deviate by about $23K on average. The spread of points increases with salary, hinting at heteroscedasticity (non-constant variance), which may impact model reliability at higher salary ranges.

Submission

1. Save figures in the `_output/` folder.
2. Commit and push code and output files:

```
git add .
git commit -m "Add Lab 08 Salary Prediction models and output"
git push origin main
```

3. Submit your GitHub repository link.

Resources

- PySpark MLlib Docs
- Seaborn Docs
- Pandas User Guide