Exploring Salary Predictions: An In-depth Analysis Using Polynomial Regression and Cross-Validation

SUBMITTED BY :

Subhash Agrawal (23CS60R67)

M. Tech - Computer Science & Engineering

Indian Institute of Technology, Khaaragpur

CS61061 - Data Analytics (Autumn 2023)

Prof. Debasis Samanta

CONTENTS

1. Project Introduction
2. Steps Involved In The Project Implementation
3. Programming code
4. Snapshot of the output
5. Results & Conclusions

PROJECT INTRODUCTION

Exploring Salary Predictions: An In-depth Analysis Using Polynomial Regression and Cross-Validation

The aim of this project is to predict the salary of an employee based on their experience in months using polynomial regression and K-fold cross validation techniques.
Polynomial regression is a type of linear regression that models the relationship between a dependent variable and one or more independent variables as a polynomial function.
K-fold cross validation is a method of splitting the data into K subsets, using one subset as the test set and the rest as the training set, and repeating this process K times. This way, we can evaluate the performance of the model on different subsets of the data and avoid overfitting or underfitting.
We will use three evaluation metrics to measure the accuracy of the model: mean squared error, R-squared, and mean absolute error. We also test the model with different polynomial degrees from 1 to 5 and compare the results. We find the best model and degree that minimizes the mean squared error, maximizes the R-squared, and minimizes the mean absolute error.
To train our model, We use the data from the Experience-Salary.csv file at (Experience Salary Dataset (kaggle.com)), which contains the experience in months and salary in thousands of rupees of the employees. We use Python libraries such as pandas, numpy, matplotlib, and sklearn to manipulate, visualize, and model the data, To choose the best model amongst them.

Project Code: DA-10
1. Identify the independent and dependent attributes.
2. Use K-Fold Cross validation method for Test and Train split of the data.
3. Use the following techniques to understand the relation between experience and salary:
 a. Linear Regression
 b. Polynomial Regression (should test with multiple polynomial degrees)
4. Use at least three different evaluation metrics for all the experiments.
5. Find the best method for the given dataset stating proper reasons.

Dataset URL :
https://www.kaggle.com/datasets/saquib7hussain/experience-salary-dataset

STEPS INVOLVED IN PROJECT IMPLEMENTATION

Step 1: Project Setup

- Import Libraries : Import necessary libraries such as pandas, numpy, matplotlib, and scikit-learn for data manipulation, analysis, and modeling.
- Define Evaluation Functions: Create functions (evaluate_model and train_and_evaluate_model) to evaluate the performance of machine learning models using different metrics like mean squared error, R-squared, and mean absolute error.

Step 2: Data Loading and Exploration

- Load the Dataset: Download the Data set from the source - Kaggle and Use pandas to read the saved(downloaded) dataset (Experience-Salary.csv) and store it in a DataFrame.

Step 3: Data Preprocessing (Identify the independent and dependent attributes.)

- Split Data into Features and Target : The independent attribute is exp(in months), which represents the experience of the employee in months. The dependent attribute is salary(in thousands), which represents the salary of the employee in thousands of rupees. Store these attributes in separate variables X and y respectively.

Step 4: Model Training and Evaluation (Using sklearn Library)

- **Use K-Fold Cross validation method for Test and Train split of the data** : Initialize K-Fold cross-validation (KFold) with the desired number of splits (say 5).
- **Use the following techniques to understand the relation between experience and salary:**
  a. Linear Regression (i.e. Polynomial Regression of degree 1)
  b. Polynomial Regression (should test with multiple polynomial degrees)

  Iterate Through Folds: - Use a loop to iterate through the folds created by K-Fold.
  In each iteration, split the data into training and testing sets.

  Polynomial Regression: - For each polynomial degree (1 to 5):
  Apply polynomial features transformation.
  Train a linear regression model.
  Evaluate the model using the defined evaluation functions.

- **Model Comparison (Use at least three different evaluation metrics for all the experiments) :**
  Compare models based on mean squared error, R-squared, and mean absolute error. Keep track of the best model and its degree

Step 5: Visualization

- **Plot Results:** Use matplotlib to create plots for R-squared, mean squared error, and mean absolute error against polynomial degree.

Step 6: Final Model Selection & Reporting (Find the best method for the given dataset stating proper reasons.)

- **Select the Best Model**: Identify the best model based on the evaluation metrics.The Model with Least Mean Squared Error , Mean Absolute Error AND Greatest R-Squared value is chosen to be the Best model
- **Print Results**: Print the best model and degree based on mean squared error, R-squared, and mean absolute error.

PROGRAMMING

CODE

- CODE WRITTEN IN PYTHON LANGUAGE (.ipynb file included in Attachment)

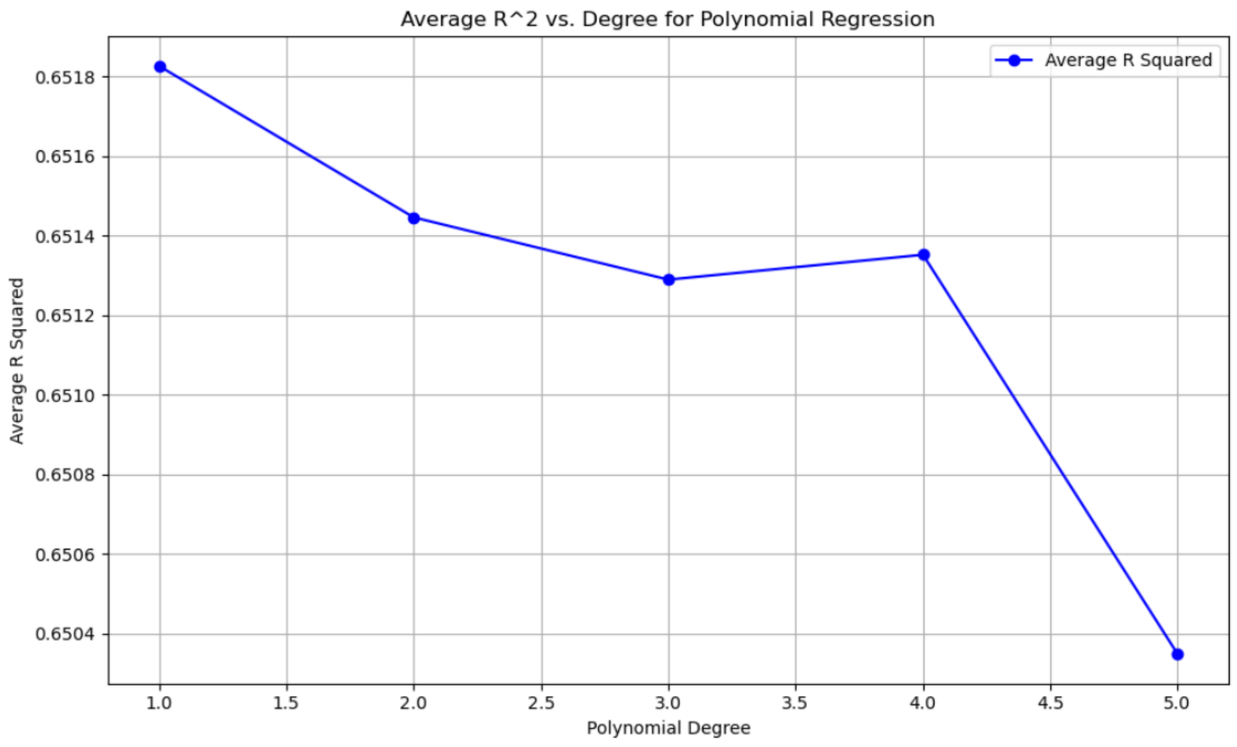
```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

#Function to Evaluate MSE, R-Squares And MAE Metrics of a model
def evaluate_model(y_true, y_pred, model_name):
    mse = mean_squared_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)

    print(f'{model_name} Results:')
    print('Mean Squared Error:', mse)
    print('R-squared:', r2)
    print('Mean Absolute Error:', mae)
    print()


#Function for Train & Evaluate the Model
def train_and_evaluate_model(X_train, X_test, y_train, y_test, model, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    evaluate_model(y_test, y_pred, model_name)

# Reading the data from the CSV file in the same folder
data = pd.read_csv('Experience-Salary.csv')
data.head()

# Identifying the independent and dependent attributes
X = data['exp(in months)'].values.reshape(-1, 1)  # Independent attribute - YearsExperience
y = data['salary(in thousands)'].values  # Dependent attribute - Salary


#Testing diffrent Models & Evaluating the Average Metrices
kf = KFold(n_splits=5, shuffle=True, random_state=42)

best_model = None
best_degree = None
best_mse = float('inf')
best_r2 = -float('inf')
best_mae = float('inf')

avg_mse_list = []
avg_r2_list = []
avg_mae_list = []
```

```
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    degrees = [1, 2, 3, 4, 5]

    mse_list = []
    r2_list = []
    mae_list = []

    for degree in degrees:
        poly = PolynomialFeatures(degree=degree)
        X_poly_train = poly.fit_transform(X_train)
        X_poly_test = poly.transform(X_test)
        pr = LinearRegression()
        train_and_evaluate_model(X_poly_train, X_poly_test, y_train, y_test, pr, f'Polynomial Regression
(degree {degree})')

        mse_pr = mean_squared_error(y_test, pr.predict(X_poly_test))
        r2_pr = r2_score(y_test, pr.predict(X_poly_test))
        mae_pr = mean_absolute_error(y_test, pr.predict(X_poly_test))

        mse_list.append(mse_pr)
        r2_list.append(r2_pr)
        mae_list.append(mae_pr)

        if mse_pr < best_mse:
            best_mse = mse_pr
            best_model = pr
            best_degree = degree

        if r2_pr > best_r2:
            best_r2 = r2_pr

        if mae_pr < best_mae:
            best_mae = mae_pr

    avg_mse_list.append(mse_list)
    avg_r2_list.append(r2_list)
    avg_mae_list.append(mae_list)

avg_mse_list = np.mean(avg_mse_list, axis=0)
avg_r2_list = np.mean(avg_r2_list, axis=0)
avg_mae_list = np.mean(avg_mae_list, axis=0)


#Visualizing the Avg. Metrices for each Degree to Conclude the best Model

# Plotting R-squared, Mean Squared Error, and Mean Absolute Error on separate subplots
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 18))
```

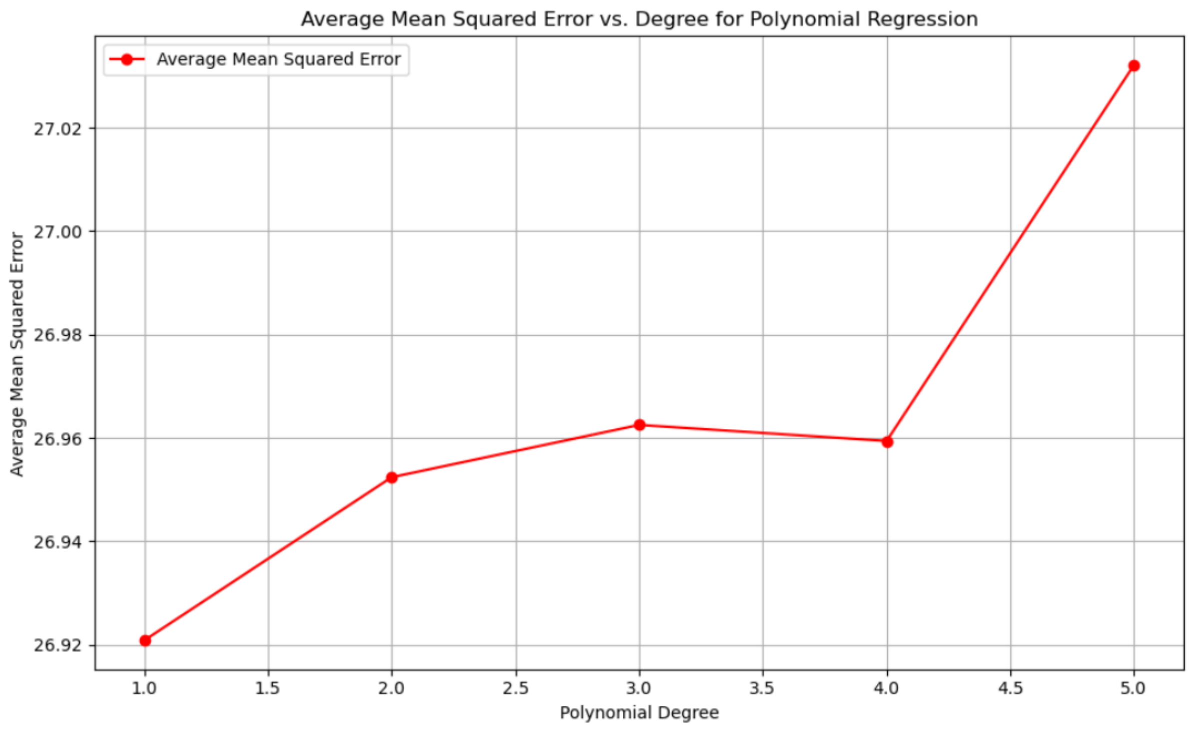
```
# Plotting R-squared
ax1.plot(degrees, avg_r2_list, marker='o', linestyle='-', color='b', label='Average R Squared')
ax1.set_title('Average R^2 vs. Degree for Polynomial Regression')
ax1.set_xlabel('Polynomial Degree')
ax1.set_ylabel('Average R Squared')
ax1.grid(True)
ax1.legend()

# Plotting Mean Squared Error
ax2.plot(degrees, avg_mse_list, marker='o', linestyle='-', color='r', label='Average Mean Squared Error')
ax2.set_title('Average Mean Squared Error vs. Degree for Polynomial Regression')
ax2.set_xlabel('Polynomial Degree')
ax2.set_ylabel('Average Mean Squared Error')
ax2.grid(True)
ax2.legend()

# Plotting Mean Absolute Error
ax3.plot(degrees, avg_mae_list, marker='o', linestyle='-', color='g', label='Average Mean Absolute Error')
ax3.set_title('Average Mean Absolute Error vs. Degree for Polynomial Regression')
ax3.set_xlabel('Polynomial Degree')
ax3.set_ylabel('Average Mean Absolute Error')
ax3.grid(True)
ax3.legend()

plt.tight_layout()
plt.show()

#Final conclusion of the BEST MODEL

# Print the best model and degree for Mean Squared Error
print(avg_mse_list)
print(f'Best Model Based on Mean Squared Error: Polynomial Regression (degree {best_degree})')
print('Reason: This model has the lowest mean squared error among all models.')
print()

# Print the best model and degree for R-squared
print(avg_r2_list)
print(f'Best Model Based on \b R Squared\b: Polynomial Regression (degree {best_degree})')
print('Reason: This model has the highest R squared among all models.')
print()

# Print the best model and degree for Mean Absolute Error
print(avg_mae_list)
print(f'Best Model Based on Mean Absolute Error: Polynomial Regression (degree {best_degree})')
print('Reason: This model has the lowest mean absolute error among all models.')
```

SNAPSHOTS OF OUTPUT

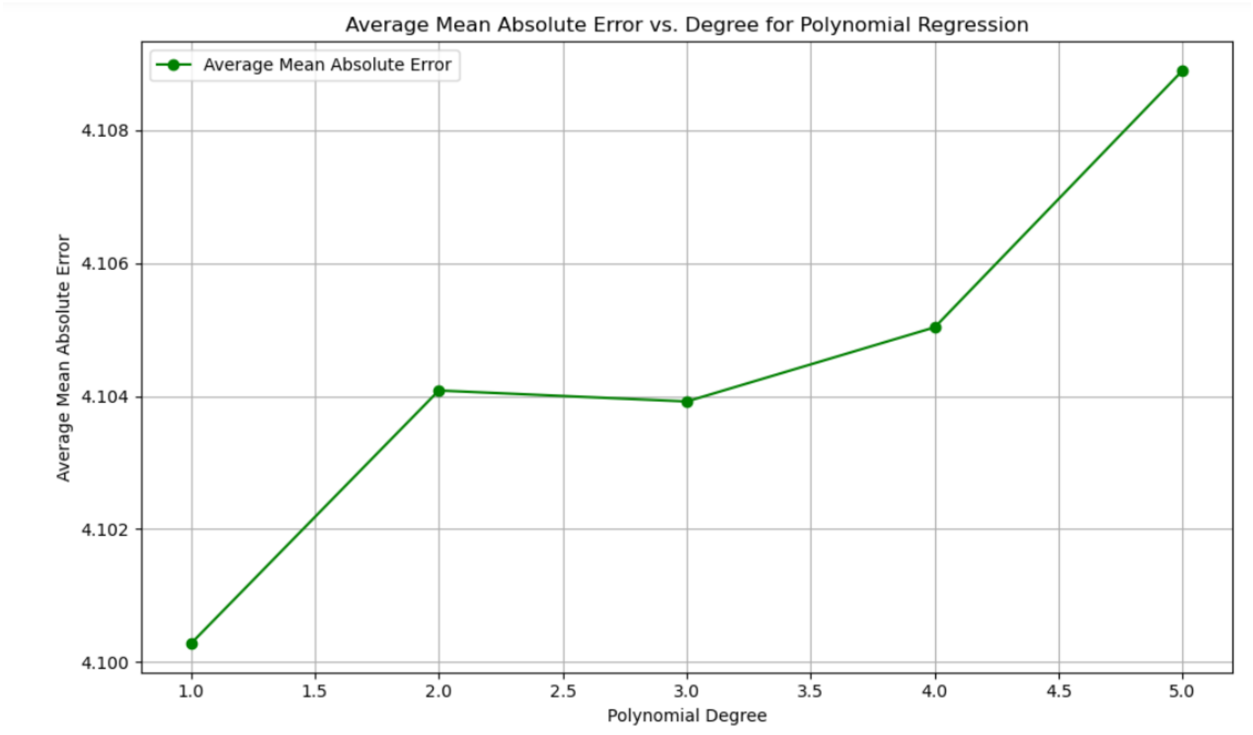
AVERAGE R-SQUARE VS DEGREE OF POLYNOMIAL



AVERAGE MEAN SQUARED ERROR (MSE) VS DEGREE OF POLYNOMIAL



AVERAGE MEAN ABSOLUTE ERROR (MAE) VS DEGREE OF POLYNOMIAL



RESULT - SNAPSHOT

```
[26.92080557 26.95238284 26.9625041  26.95939619 27.03217554]
Best Model Based on Mean Squared Error: Polynomial Regression (degree 1)
Reason: This model has the lowest mean squared error among all models.

[0.65182588 0.65144542 0.65128894 0.65135143 0.6503478 ]
Best Model Based on R Square: Polynomial Regression (degree 1)
Reason: This model has the highest R squared among all models.

[4.10027775 4.10408363 4.10391868 4.10503429 4.10889713]
Best Model Based on Mean Absolute Error: Polynomial Regression (degree 1)
Reason: This model has the lowest mean absolute error among all models.
```

RESULTS & CONCLUSION

Based on the Metrics Obtained we can Conclude that the Linear Regression (i.e. Polynomial Regression with degree 1) is best suited, to train the Model over this Data, to predict Salary based on Experience As

1. It has Least Mean Squared Error
2. It has Least Mean Absolute Error
3. It has Greatest R-squared Value.