dataset: https://www.kaggle.com/datasets/lovishbansal123/engineering-graduate-salary

Description:

This dataset offers an in-depth examination of salary trends among engineering graduates, covering a wide range of variables including the engineering specialty, alma mater, graduation year, job location, employer, position held, initial and current salaries, and industry tenure. The dataset's purpose is to facilitate the analysis of salary patterns and trends for those with engineering degrees, ensuring all personal data is anonymized to maintain individual confidentiality.

The dataset has 34 fields:

ID: A unique identifier for each graduate. Gender: The gender of the graduate (f for female, m for male).

DOB: Date of birth of the graduate.

10percentage: Percentage obtained in 10th grade.

10board: The board of education for the 10th grade.

12graduation: The year of graduation from 12th grade.

12percentage: Percentage obtained in 12th grade.

12board: The board of education for the 12th grade.

CollegeID: A unique identifier for the college from which the graduate obtained their degree.

CollegeTier: Tier of the college (1 for top tier colleges, 2 for others).

Degree: The type of degree obtained (e.g., B.Tech, M.Tech).

Specialization: The engineering specialization (e.g., Computer Science, Electrical). CollegeGPA: Grade point average in college.

CollegeCityID: A unique identifier for the city of the college.

CollegeCityTier: Tier of the college city (1 for metros, 2 for others).

CollegeState: The state where the college is located.

GraduationYear: The year of graduation from college.

English: Score in an English language assessment.

Logical: Score in a logical reasoning assessment.

Quant: Score in a quantitative ability assessment.

Domain: A score representing the domain knowledge.

ComputerProgramming: Score in a computer programming test.

ElectronicsAndSemicon: Score in an electronics and semiconductor engineering test.

ComputerScience: Score in a computer science test.

MechanicalEngg: Score in a mechanical engineering test.

ElectricalEngg: Score in an electrical engineering test.

TelecomEngg: Score in a telecommunications engineering test.

CivilEngg: Score in a civil engineering test.

conscientiousness, agreeableness, extraversion, nueroticism, openess_to_experience: These fields represent personality traits measured via standardized psychological tests.

Salary: The current salary of the graduate.

```
# Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm



data_file = 'Engineering_graduate_salary.csv'
data= pd.read_csv(data_file)


# Display the first 10 rows of the dataset
print(data.head(10))
```

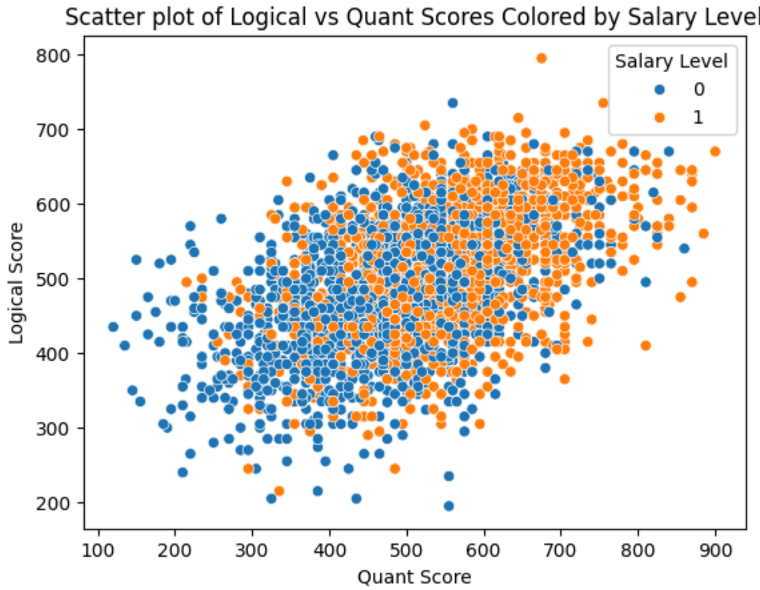
```
        ID Gender         DOB  10percentage                      10board  \
0   604399      f  1990-10-22         87.80                         cbse
1   988334      m  1990-05-15         57.00                         cbse
2   301647      m  1989-08-21         77.33  maharashtra state board,pune
3   582313      m  1991-05-04         84.30                         cbse
4   339001      f  1990-10-30         82.00                         cbse
5   609356      f  1989-12-02         83.16                         icse
6  1081649      f  1989-04-17         72.50                  state board
7   610842      f  1991-04-11         77.00                  state board
8  1183070      m  1992-11-25         76.80                  state board
9   794062      f  1993-03-15         57.00                  state board

   12graduation  12percentage                   12board  CollegeID  \
0          2009         84.00                      cbse       6920
1          2010         64.50                      cbse       6624
2          2007         85.17  amravati divisional board       9084
3          2009         86.00                      cbse       8195
4          2008         75.00                      cbse       4889
5          2007         77.00                      cbse      10950
6          2007         53.20               state board      14381
7          2009         88.00               state board      13208
8          2010         87.70               state board       5338
9          2009         73.00               state board       8346

   CollegeTier  ... MechanicalEngg ElectricalEngg  TelecomEngg  CivilEngg  \
0            1  ...             -1             -1           -1         -1
1            2  ...             -1             -1           -1         -1
2            2  ...             -1             -1          260         -1
3            1  ...             -1             -1           -1         -1
4            2  ...             -1             -1           -1         -1
5            1  ...             -1             -1          313         -1
6            2  ...            469             -1           -1         -1
7            2  ...             -1             -1           -1         -1
8            2  ...             -1             -1           -1         -1
9            2  ...             -1             -1           -1         -1

   conscientiousness agreeableness  extraversion  nueroticism  \
0            -0.1590        0.3789        1.2396      0.14590
1             1.1336        0.0459        1.2396      0.52620
2             0.5100       -0.1232        1.5428     -0.29020
3            -0.4463        0.2124        0.3174      0.27270
4            -1.4992       -0.7473       -1.0697      0.06223
5             0.8463       -0.6201       -0.7585     -0.99500
6             0.1282       -0.4536        0.3174      0.90660
7             0.1282        0.5454        0.4711      0.90660
8            -0.1590       -0.4536        0.1637      0.52620
9            -0.7335       -0.4536       -0.2974      1.41360

   openess_to_experience  Salary
0                 0.2889  445000
1                -0.2859  110000
2                -0.2875  255000
3                 0.4805  420000
4                 0.1864  200000
5                -0.2859  440000
6                -0.0943  150000
7                -0.2859  105000
8                -0.0943  195000
```

```
# Show the DataFrame information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2998 entries, 0 to 2997
Data columns (total 34 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     2998 non-null   int64
 1   Gender                 2998 non-null   object
 2   DOB                    2998 non-null   object
 3   10percentage           2998 non-null   float64
 4   10board                2998 non-null   object
 5   12graduation           2998 non-null   int64
 6   12percentage           2998 non-null   float64
 7   12board                2998 non-null   object
 8   CollegeID              2998 non-null   int64
 9   CollegeTier            2998 non-null   int64
 10  Degree                 2998 non-null   object
 11  Specialization         2998 non-null   object
 12  collegeGPA             2998 non-null   float64
 13  CollegeCityID          2998 non-null   int64
 14  CollegeCityTier        2998 non-null   int64
 15  CollegeState           2998 non-null   object
 16  GraduationYear         2998 non-null   int64
 17  English                2998 non-null   int64
 18  Logical                2998 non-null   int64
 19  Quant                  2998 non-null   int64
 20  Domain                 2998 non-null   float64
 21  ComputerProgramming    2998 non-null   int64
 22  ElectronicsAndSemicon  2998 non-null   int64
 23  ComputerScience        2998 non-null   int64
 24  MechanicalEngg         2998 non-null   int64
 25  ElectricalEngg         2998 non-null   int64
 26  TelecomEngg            2998 non-null   int64
 27  CivilEngg              2998 non-null   int64
 28  conscientiousness      2998 non-null   float64
 29  agreeableness          2998 non-null   float64
 30  extraversion           2998 non-null   float64
 31  nueroticism            2998 non-null   float64
 32  openess_to_experience  2998 non-null   float64
 33  Salary                 2998 non-null   int64
dtypes: float64(9), int64(18), object(7)
memory usage: 796.5+ KB
```

```
# Prepare data for logistic regression

# Calculate the median salary
median_salary = data['Salary'].median()

# Create a binary outcome variable where 1 indicates a salary above the median
data['HighSalary'] = (data['Salary'] > median_salary).astype(int)

# Define features and target variable
X = data[['English', 'Logical', 'Quant']]
y = data['HighSalary']

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Visualize the data
sns.scatterplot(data=data, x='Quant', y='Logical', hue='HighSalary')
plt.title('Scatter plot of Logical vs Quant Scores Colored by Salary Level')
plt.xlabel('Quant Score')
plt.ylabel('Logical Score')
plt.legend(title='Salary Level')
plt.show()
```




```
# Initialize and fit the logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Display the coefficients
print("Coefficients:", logreg.coef_)
print("Intercept:", logreg.intercept_)
```

```
Coefficients: [[0.00331005 0.0017523  0.00519913]]
Intercept: [-5.40105585]
```

```
# Selecting the first row from the test set, modifying it, and making a prediction
sample_data = X_test.iloc[0].values.reshape(1, -1)

# Make a prediction
prediction = logreg.predict(sample_data)
print("Predicted Class:", prediction)
```

```
Predicted Class: [1]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticR
  warnings.warn(
```

```
# Set up the data for LDA
X = data[['English', 'Logical', 'Quant']]
y = data['HighSalary']
```

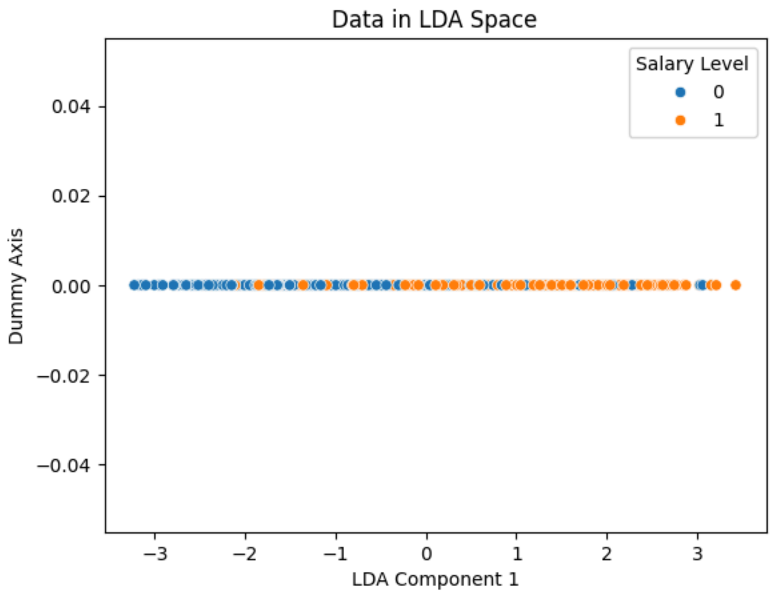
```
# Initialize the LDA model
lda = LinearDiscriminantAnalysis()

# Fit the model to your data
lda.fit(X_train, y_train)
```

```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

```
# Print the LDA variance ratio
print("Variance Ratios:", lda.explained_variance_ratio_)
```

```
Variance Ratios: [1.]
```

```
# Transform the data
X_lda = lda.transform(X)


# LDA reduces to 1 component for simplicity in visualization
sns.scatterplot(x=X_lda[:, 0], y=[0]*len(X_lda), hue=y)
plt.title('Data in LDA Space')
plt.xlabel('LDA Component 1')
plt.ylabel('Dummy Axis')
plt.legend(title='Salary Level')
plt.show()
```




```
# Set up the data for K-fold cross validation using linear regression
X = data[['English', 'Logical', 'Quant']]
y = data['Salary']



# Initialize the linear regression model
lin_reg = LinearRegression()



# Define a 5-fold cross-validation split
kf = KFold(n_splits=5, random_state=42, shuffle=True)


# Initialize a list to store the mean squared errors for each fold
mse_scores = []

# Manually loop through each fold
for train_index, test_index in kf.split(X):
    # Split the data into training and testing sets for the current fold
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit the model to the training data
    lin_reg.fit(X_train, y_train)

    # Make predictions on the testing data
    y_pred = lin_reg.predict(X_test)

    # Calculate the mean squared error for the current fold
    mse = mean_squared_error(y_test, y_pred)
    mse_scores.append(mse)
```

```
# Print the MSE for each fold
for i, score in enumerate(mse_scores, 1):
    print(f"Fold {i}: Mean Squared Error: {score}")

# Print the average MSE
print(f"Average Mean Squared Error: {sum(mse_scores) / len(mse_scores)}")
```

```
Fold 1: Mean Squared Error: 27116441704.79011
Fold 2: Mean Squared Error: 66778311682.00544
Fold 3: Mean Squared Error: 51265270183.03146
Fold 4: Mean Squared Error: 47587654177.49975
Fold 5: Mean Squared Error: 18561177636.584328
Average Mean Squared Error: 42261771076.78222
```

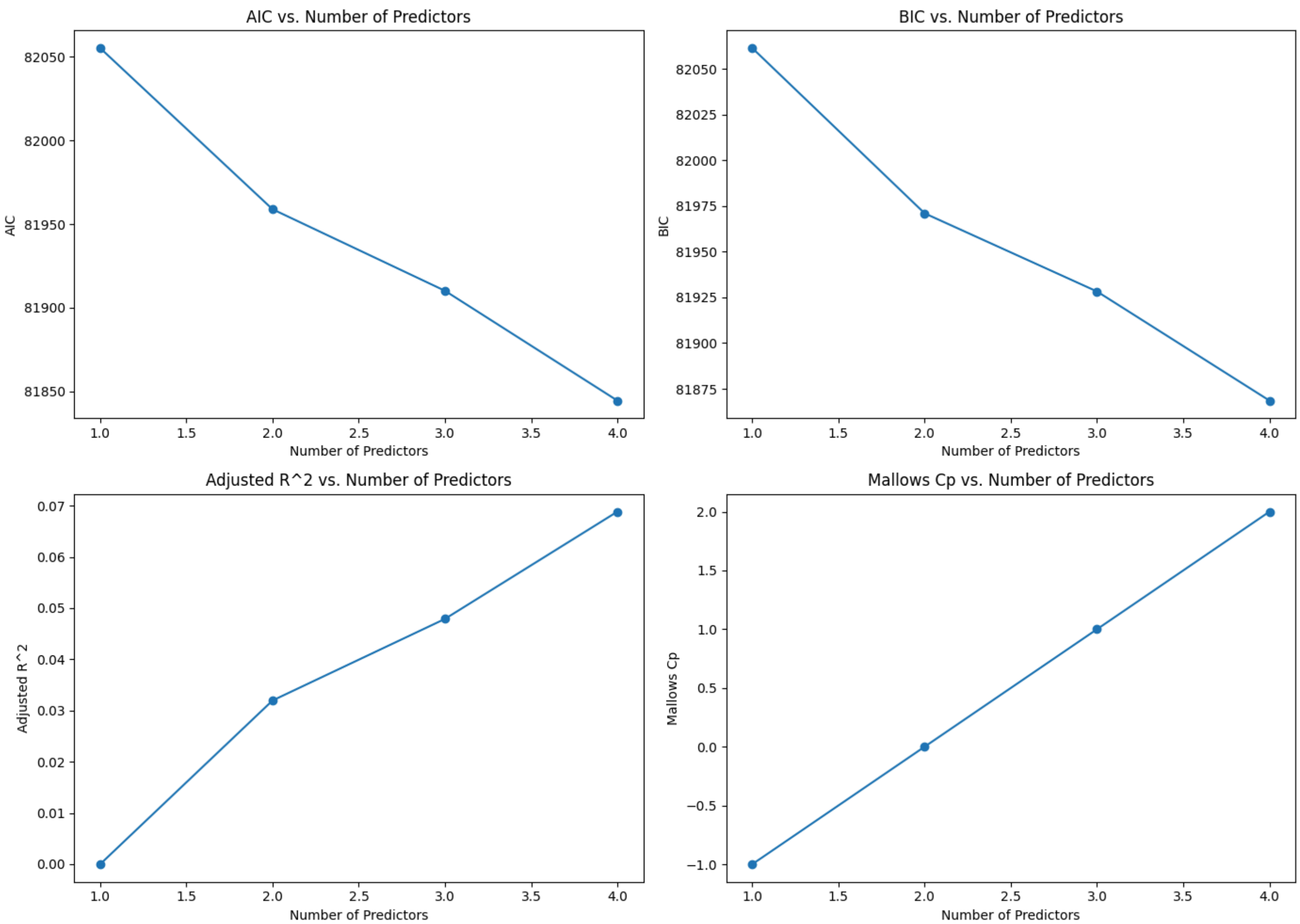
```
# Define features and target
X = data[['English', 'Logical', 'Quant']]
y = data['Salary']

# Add a constant to the feature matrix to represent the intercept
X = sm.add_constant(X)
```

```
def mallows_cp(model, y_actual):
    rss = sum(model.resid ** 2)
    p = model.df_model  # Number of predictors
    n = len(y_actual)  # Total number of observations
    mse = rss / (n - p - 1)
    cp = rss / mse - (n - 2 * p)
    return cp
```

```
# Initialize lists to store the metrics for each model
aic_list = []
bic_list = []
adj_r_squared_list = []
cp_list = []

# Loop over models with an increasing number of predictors
for i in range(1, X.shape[1] + 1):
    # Select the first i columns from X
    Xi = X.iloc[:, :i]

    # Fit the model
    model = sm.OLS(y, Xi).fit()

    # Store the metrics
    aic_list.append(model.aic)
    bic_list.append(model.bic)
    adj_r_squared_list.append(model.rsquared_adj)
    cp_list.append(mallows_cp(model, y))
```

```
# Plot AIC, BIC, Adjusted R^2, and Cp
plt.figure(figsize=(14, 10))

# Plot AIC
plt.subplot(2, 2, 1)
plt.plot(range(1, len(aic_list) + 1), aic_list, marker='o')
plt.xlabel('Number of Predictors')
plt.ylabel('AIC')
plt.title('AIC vs. Number of Predictors')

# Plot BIC
plt.subplot(2, 2, 2)
plt.plot(range(1, len(bic_list) + 1), bic_list, marker='o')
plt.xlabel('Number of Predictors')
plt.ylabel('BIC')
plt.title('BIC vs. Number of Predictors')

# Plot Adjusted R^2
plt.subplot(2, 2, 3)
plt.plot(range(1, len(adj_r_squared_list) + 1), adj_r_squared_list, marker='o')
plt.xlabel('Number of Predictors')
plt.ylabel('Adjusted R^2')
plt.title('Adjusted R^2 vs. Number of Predictors')
```

```
# Plot Cp
plt.subplot(2, 2, 4)
plt.plot(range(1, len(cp_list) + 1), cp_list, marker='o')
plt.xlabel('Number of Predictors')
plt.ylabel('Mallows Cp')
plt.title('Mallows Cp vs. Number of Predictors')

plt.tight_layout()
plt.show()
```




```
# Define the predictor features and the target variable
features = ['collegeGPA', 'English', 'Logical', 'Quant']
X = data[features]
y = data['Salary']
```

```
# Create a 10th-degree PolynomialFeatures object
poly_features_10 = PolynomialFeatures(degree=10, include_bias=False)
X_train_poly_10 = poly_features_10.fit_transform(X_train)
X_test_poly_10 = poly_features_10.transform(X_test)

# Create a 4th-degree PolynomialFeatures object
poly_features_4 = PolynomialFeatures(degree=4, include_bias=False)
X_train_poly_4 = poly_features_4.fit_transform(X_train)
X_test_poly_4 = poly_features_4.transform(X_test)

# Create a linear regression model and train it on the transformed features
model_4 = LinearRegression()
model_4.fit(X_train_poly_4, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Create a 3rd-degree PolynomialFeatures object
poly_features_3 = PolynomialFeatures(degree=3, include_bias=False)
X_train_poly_3 = poly_features_3.fit_transform(X_train)
X_test_poly_3 = poly_features_3.transform(X_test)

# Create a linear regression model and train it on the transformed features
model_3 = LinearRegression()
model_3.fit(X_train_poly_3, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# Calculate and print MSE for the 10th-degree polynomial model
y_train_pred_10 = model_10.predict(X_train_poly_10)
y_test_pred_10 = model_10.predict(X_test_poly_10)
print(f"10-degree Polynomial Regression - MSE Train: {mean_squared_error(y_train, y_train_pred_10)}, MSE Test: {mean_squared_err

# Calculate and print MSE for the 4th-degree polynomial model
y_train_pred_4 = model_4.predict(X_train_poly_4)
y_test_pred_4 = model_4.predict(X_test_poly_4)
print(f"4-degree Polynomial Regression - MSE Train: {mean_squared_error(y_train, y_train_pred_4)}, MSE Test: {mean_squared_error

# Calculate and print MSE for the 3rd-degree polynomial model
y_train_pred_3 = model_3.predict(X_train_poly_3)
y_test_pred_3 = model_3.predict(X_test_poly_3)
print(f"3-degree Polynomial Regression - MSE Train: {mean_squared_error(y_train, y_train_pred_3)}, MSE Test: {mean_squared_error
```

```
10-degree Polynomial Regression - MSE Train: 43048304097.73153, MSE Test: 121994931553399.74
4-degree Polynomial Regression - MSE Train: 46694019213.74283, MSE Test: 20415893746.486206
3-degree Polynomial Regression - MSE Train: 47133337220.413765, MSE Test: 19220062523.35299
```


[Nikita Belii](#)
7:40pm

Analyzing the engineers' salaries data, I noticed some interesting patterns. The quantitative score is the most significant factor in predicting whether a graduate earns a higher salary, as shown in the initial regression analysis. The scatter plot suggests a pattern where those with higher quantitative scores often have higher salaries, although it's not a clear-cut distinction. The LDA analysis confirmed that separating high and low earners isn't straightforward with the data at hand. When I checked the model's reliability with k-fold cross-validation, I saw some inconsistency in its predictions depending on how the data was split, suggesting the model could be fine-tuned further. With polynomial regression, the simpler models with fewer variables were more predictive than more complex ones, which seemed to fit the data too closely. Models with a lower number of predictors were generally better, as indicated by statistics like AIC and BIC. All in all, the study shows that while certain academic scores can give us clues about salary expectations, the reality is likely affected by a combination of many factors

Reply