Term Project

Samuel Castro

Bellevue University - M.S Data Science

DSC 630 – Predictive Analytics

Professor Andrew Hua

July 17, 2023

Predicting Salary

Samuel Castro

Milestone 3 -Preliminary Analysis

Data Potential

The data I acquired is complete in the sense that there are no missing values and that there are no issues with the data itself. I will be using three variables in my model. Level of experience, salary, and job type. I can add in location to make the prediction model more intricate. One issue is the categorical variables. One way to handle categorical variables is to use one-hot encoding. For example, if you have a "job type" variable with categories like "data analyst," "data scientist," and "data engineer," I would create three separate binary variables (e.g., "is_data_analyst," "is_data_scientist," and "is_data_engineer") with values of 0 or 1 indicating the presence or absence of each category. The target variable is salary, and the most correlated variable is level of experience. It appears level of experience affects salary the most, with job type only vaguely affecting salary. I believe that with the data I have acquired I can make useful predictions. Location seems to be a useful variable and I will add it in the model after I create the initial prediction model. I plan on investigating whether there is a salary increase as experience grows, and if certain job types tend to offer higher salaries compared to others.

Motivation:

My main motivation for this term project and building this model is to see what I am going to go through when I graduate. I hope that this prediction model will help ease my stress, but I also aim to give companies the ability to make data driven decisions in terms of compensation, career planning, and talent acquisition. Predicting salary is a great way of showing people what they

will be making after they apply and why they should be earning that. I want to create a baseline prediction model that will show applicants what they will be making in the area they are applying to.

Visualizations:

Visualizations that can be particularly useful in my case include scatter plots, box plots, and bar charts. Scatter plots can show the relationship between experience and salary, while box plots and bar charts can provide insights into salary distributions across different job types and experience levels. There are no outliers in my data so there is no need to emphasize the outliers in graphs. A correlation matrix will also be useful to show the level of correlation between the variables.

Model Expectations:

I believe that multiple linear regression and decision tree regression is the best choice for this model because these models can capture both linear and non-linear relationships between job type, level of experience, and salary. I will also add job location to a model to see if this affects salary and the model that we are building. By incorporating job type, level of experience, and job location as predictors, you can explore their impact on salary. My expectations are reasonable because I have enough data to predict the salary range.

Milestone 2 Project Proposal:

“Information is the oil of the 21st century, and analytics is the combustion engine.”

~ Peter Sondergaard, Senior Vice President and Global Head of Research at Gartner, Inc

I got into data science because in my undergrad I knew that data was the future. Information, and technology runs the world. My professors beat that into our heads. They said that you will not go wrong with data. As I currently complete my masters, I feel a sense of fear and excitement as I near the end. I start to wonder about how the pay is, which is why for my predictive analytic project I am going to develop a model that estimates the salary of data professionals based on their level of experience and job type. Understanding the factors influencing data professionals' salaries is crucial for various stakeholders, including job seekers, employers, HR departments, and industry analysts, and myself. At the end of this project, I hope to alleviate some of the dread that is building up inside of me. I also aim to give companies the ability to make data driven decisions in terms of compensation, career planning, and talent acquisition.

My data was selected from Kaggle.com. I researched data sites that would give me level of experience, job type and salary in a data related job. There are many factors that come into play when considering a hiring process, but I am going to narrow the field to just level of experience, which ranges from senior, to mid, to entry level, and job type. Job type ranges from full time and internship. The first goal of this project to develop a model to provide insights into the key drivers of salary variations in the data industry.

For this project, I plan to use multiple regression models as they are well-suited for estimating salaries based on multiple features.

Specifically, I will explore the following models:

1. Multiple Linear Regression
2. Decision Tree Regression
3. Random Forest Regression

These models can capture both linear and non-linear relationships between job type, level of experience, and salary. I will also add job location to a model to see if this affects salary and the model that we are building.

To evaluate the results of the models, I will use evaluation metrics such as mean absolute error (MAE) and mean squared error (MSE). These metrics will help me calculate the prediction accuracy by measuring the average difference between the predicted and actual salary values. I will also consider R-squared (coefficient of determination) to assess the variance explained by the models.

Through this project, I hope to learn the following:

1. The relative importance of job type and experience level in determining data professionals' salaries.
2. The level of accuracy achievable in predicting salaries based on job type and experience level, and location.
3. The potential non-linear relationships and interactions between job type, experience level, salary, and location
4. The insights and patterns that emerge from the analysis, such as salary growth rates.
5. I will be understanding how entry level jobs are paid as a data scientist.

I truly am excited to unpack what this dataset is all about and just how much I can predict using only a couple of variables. I hope to use these variables to their maximum potential and

hopefully not have to find other datasets to add. If adding a dataset is necessary then I will combine the data to create one readable datafile.

Regarding ethics, I believe that my project does not have many ethical implications or risks. Some risks may include the presence of bias in the data, such as gender or racial issues in salaries. These are potential risks but the data I am handling does not include gender or racial variables. Another risk is bias, and it is crucial to watch bias level during data collection and ensure fairness and equity in the analysis. I am biased because I would love for overall results to be successful because I am going into the data industry. Additionally, privacy and data security should be maintained throughout this project, obtaining the data through the correct sites and correct methods.

In case my original plan does not work than I will have several contingencies in place. This may include exploring additional data sources to augment the existing dataset, incorporating other relevant features such as education level or certifications, or considering different models that may better capture the complexities of the salary prediction problem. Overall, I need to be flexible and adaptable to the project so I can fully flesh out the problem statement and have a valuable model outcome.

It is important to acknowledge that salary is influenced by various factors beyond job type and experience level, such as industry, location, and skills. I recognize that this project is limited because of the number of variables I am using. The foundation is crucial to a data science project. It is essential to be flexible with your process and data, but your foundation needs to be watertight because your foundation is what you are testing and building your model around.

Data

https://www.kaggle.com/datasets/milanvaddoriya/data-science-job-salary

Samuel Castro

8.3 Course Project: Milestone 4--Finalizing Your Results

In [1]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

In [2]:
```python
# Load the data from txt file
file_path = 'Desktop/DSC 630/datascience_salaries.csv'
df = pd.read_csv(file_path)
```

In [3]:
```python
df.head()
```

Out[3]:

	Unnamed: 0	job_title	job_type	experience_level	location	salary_currency	salary
0	0	Data scientist	Full Time	Senior	New York City	USD	149000
1	2	Data scientist	Full Time	Senior	Boston	USD	120000
2	3	Data scientist	Full Time	Senior	London	USD	68000
3	4	Data scientist	Full Time	Senior	Boston	USD	120000
4	5	Data scientist	Full Time	Senior	New York City	USD	149000

Explain your process for prepping the data:

The process of prepping the data involves cleaning, transforming, and preparing the dataset for analysis and model building.

In this case, I started by loading the provided data into a pandas DataFrame.

I then performed some simple cleaning steps where I removed any duplicate rows and checked for missing values. I then did a simple check of outliers by checking the maximum and minimum values for salary.

I identified the target variable (salary) and the predictor variables (job_type, experience_level, and location).

Since the dataset contained categorical variables (job_type, experience_level, and location), I used one-hot encoding to convert them into a numerical format suitable for regression models.

I split the data into training and testing sets to train the models on one portion of the data and evaluate their performance on another independent portion.

In [4]:
```python
# Remove duplicate rows, if any
df.drop_duplicates(inplace=True)
```

In [5]:
```python
# Check for missing values in the data
missing_values = df.isnull().sum()

# Display the count of missing values for each column
print(missing_values)
```

```
Unnamed: 0          0
job_title           0
job_type            0
experience_level    0
location            0
salary_currency     0
salary              0
dtype: int64
```

In [6]:
```python
# Find the maximum and minimum salary values
max_salary = df['salary'].max()
min_salary = df['salary'].min()

# Display the results
print("Maximum Salary:", max_salary)
print("Minimum Salary:", min_salary)
```

```
Maximum Salary: 228000
Minimum Salary: 30000
```

In [7]:
```python
# Step 1: Data Preprocessing - Convert categorical variables into numerical values
# I will use one-hot encoding for job_title, job_type, experience_level, and location

categorical_columns = ['job_title', 'job_type', 'experience_level']
numeric_columns = ['salary']

# One-hot encode the categorical columns
encoder = OneHotEncoder(drop='first', sparse=False)
encoded_features = encoder.fit_transform(df[categorical_columns])

# Create a new DataFrame with the encoded features and the numeric columns
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_columns))
data_processed = pd.concat([encoded_df, df[numeric_columns]], axis=1)
```

The first model I will be building is going to be multiple linear regression and the second model will be decision tree regression. The reason I am using both of these models is because these models can capture both linear and non-linear relationships between job type, level of experience, and salary.

In [8]:
```python
# Step 2: Split the data into training and testing sets
X = data_processed.drop('salary', axis=1)
y = data_processed['salary']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [9]:
```python
# Step 3: Build the Multiple Linear Regression model
linear_regression_model = LinearRegression()
linear_regression_model.fit(X_train, y_train)
```

Out[9]:
```
LinearRegression()
```

In [10]:
```python
# Step 4: Build the Decision Tree Regression model
decision_tree_model = DecisionTreeRegressor(random_state=42)
decision_tree_model.fit(X_train, y_train)
```

Out[10]:
```
DecisionTreeRegressor(random_state=42)
```

In [11]:
```python
# Step 5: Evaluate the models
# For the linear regression model
y_pred_linear = linear_regression_model.predict(X_test)
mse_linear = mean_squared_error(y_test, y_pred_linear)
print(f"Multiple Linear Regression Mean Squared Error: {mse_linear}")

# For the decision tree regression model
y_pred_tree = decision_tree_model.predict(X_test)
mse_tree = mean_squared_error(y_test, y_pred_tree)
print(f"Decision Tree Regression Mean Squared Error: {mse_tree}")
```

```
Multiple Linear Regression Mean Squared Error: 1129852350.874494
Decision Tree Regression Mean Squared Error: 1119725328.3491523
```

The MSE for the Multiple Linear Regression model is approximately 1,129,852,350. This value represents the average squared difference between the predicted salary values by the Multiple Linear Regression model and the actual salary values in the dataset. Since the MSE is relatively high, it suggests that the model has some level of error in its predictions, meaning it's not perfectly accurate in estimating the salaries. However, the MSE value is still reasonable, indicating that the Multiple Linear Regression model is making predictions that are fairly close to the actual salary values.

The MSE for the Decision Tree Regression model is approximately 1,119,725,328. The MSE for the Decision Tree Regression model is approximately 1,119,725,328.35. This value represents the average squared difference between the predicted salary values by the Decision Tree Regression model and the actual salary values in the dataset. Similar to the Multiple Linear Regression model, the Decision Tree model also has some level of error in its predictions. However, the Decision Tree model's MSE is slightly lower compared to the Multiple Linear Regression model, indicating that the Decision Tree model is making slightly more accurate predictions on average.

Both models are performing relatively similarly with the Decision Tree Regression model having a slightly lower MSE. This suggests that the Decision Tree model is slightly better at making predictions on this specific dataset compared to the Multiple Linear Regression model. However, it's important to note that the difference in MSE values is not substantial, and further analysis or fine-tuning of the models may be needed to improve prediction accuracy.

In the next models I will add location to see if it plays a part in the prediction model

In [12]:
```python
# Step 1: Data Preprocessing - Convert categorical variables into numerical values
# I will use one-hot encoding for job_title, job_type, experience_level, and location

categorical_columns1 = ['job_title', 'job_type', 'experience_level', 'location']
numeric_columns1 = ['salary']

# One-hot encode the categorical columns
encoder = OneHotEncoder(drop='first', sparse=False)
encoded_features = encoder.fit_transform(df[categorical_columns1])

# Create a new DataFrame with the encoded features and the numeric columns
encoded_df1 = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_columns1))
data_processed1 = pd.concat([encoded_df, df[numeric_columns1]], axis=1)
```

In [13]:
```python
# Step 2: Split the data into training and testing sets
X1 = data_processed1.drop('salary', axis=1)
y1 = data_processed1['salary']
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.2, random_state=42)
```

In [14]:
```python
# Step 3: Build the Multiple Linear Regression model
linear_regression_model1 = LinearRegression()
linear_regression_model1.fit(X_train1, y_train1)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```python
# Step 4: Build the Decision Tree Regression model
decision_tree_model1 = DecisionTreeRegressor(random_state=42)
decision_tree_model1.fit(X_train1, y_train1)
```

Out[15]:
```
DecisionTreeRegressor(random_state=42)
```

In [16]:
```python
# Step 5: Evaluate the models
# For the linear regression model
y_pred_linear1 = linear_regression_model1.predict(X_test1)
mse_linear1 = mean_squared_error(y_test1, y_pred_linear1)
print(f"Multiple Linear Regression Mean Squared Error: {mse_linear1}")

# For the decision tree regression model
y_pred_tree1 = decision_tree_model1.predict(X_test1)
mse_tree1 = mean_squared_error(y_test1, y_pred_tree1)
print(f"Decision Tree Regression Mean Squared Error: {mse_tree}")
```

```
Multiple Linear Regression Mean Squared Error: 1129852350.874494
Decision Tree Regression Mean Squared Error: 1119725328.3491523
```

I will perform further analysis of the models to improve prediction accuracy in the next code because the prediction accuracy still did not improve with adding location.

In this updated code, we have introduced feature engineering by creating an interaction term between the 'job_title_ML Ops' and 'experience_level_Senior' features. Additionally, we've standardized the numerical features using StandardScaler.

Moreover, for the Multiple Linear Regression model, we've switched to using Ridge regression (L2 regularization) by setting alpha=1.0. The alpha parameter controls the strength of regularization, and you can experiment with different values to find the one that works best for your data.

In [17]:
```python
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import Ridge  # Use Ridge for regularization
```

In [18]:
```python
# Step 1: Data Preprocessing - Convert categorical variables into numerical values
categorical_columns = ['job_title', 'job_type', 'experience_level']
numeric_columns = ['salary']

# One-hot encode the categorical columns
encoder = OneHotEncoder(drop='first', sparse=False)
encoded_features = encoder.fit_transform(df[categorical_columns])

# Create a new DataFrame with the encoded features and the numeric columns
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_columns))
data_processed = pd.concat([encoded_df, df[numeric_columns]], axis=1)
```

In [19]:
```python
# Step 2: Feature Engineering - Create interaction terms and standardize numeric columns
# For simplicity, I will create an interaction term between 'job_title_ML Ops' and 'experience_level_Senior'
data_processed['job_title_ML Ops * experience_level_Senior'] = data_processed['job_title_ML Ops'] * data_processed['experience_level_Senior']

# Standardize numeric columns
scaler = StandardScaler()
data_processed[numeric_columns] = scaler.fit_transform(data_processed[numeric_columns])
```

In [20]:
```python
# Step 3: Split the data into training and testing sets
X = data_processed.drop('salary', axis=1)
y = data_processed['salary']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [21]:
```python
# Step 4: Build the Multiple Linear Regression model with Ridge regularization
ridge_regression_model = Ridge(alpha=1.0)  # Use alpha value as needed for regularization strength
ridge_regression_model.fit(X_train, y_train)
```

Out[21]:
```
Ridge()
```

In [22]:
```python
# Step 5: Build the Decision Tree Regression model
decision_tree_model = DecisionTreeRegressor(random_state=42)
decision_tree_model.fit(X_train, y_train)
```

Out[22]:
```
DecisionTreeRegressor(random_state=42)
```

In [23]:
```python
# Step 6: Evaluate the models
# For the Ridge Regression model
y_pred_ridge = ridge_regression_model.predict(X_test)
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
print(f"Ridge Regression Mean Squared Error: {mse_ridge}")

# For the Decision Tree Regression model
y_pred_tree = decision_tree_model.predict(X_test)
mse_tree = mean_squared_error(y_test, y_pred_tree)
print(f"Decision Tree Regression Mean Squared Error: {mse_tree}")
```

```
Ridge Regression Mean Squared Error: 1.0696236598014985
Decision Tree Regression Mean Squared Error: 1.0579698649783258
```

As you can see the model is working significantly better when compared to the first couple of models I ran

The MSE for the Ridge Regression model is approximately 1.07. This value represents the average squared difference between the predicted salary values by the Ridge Regression model and the actual salary values in the dataset. Since the MSE is relatively low, it suggests that the Ridge Regression model is making accurate predictions, and the predicted salary values are quite close to the actual salary values in the dataset.

The MSE for the Decision Tree Regression model is approximately 1.06. This value represents the average squared difference between the predicted salary values by the Decision Tree Regression model and the actual salary values in the dataset. Similar to the Ridge Regression model, the Decision Tree model also has a relatively low MSE, indicating that it is making accurate predictions, and the predicted salary values are close to the actual salary values in the dataset.

Repeating the process but with location added in

In [24]:
```python
# Step 1: Data Preprocessing - Convert categorical variables into numerical values
categorical_columns = ['job_title', 'job_type', 'experience_level', 'location']
numeric_columns = ['salary']

# One-hot encode the categorical columns
encoder = OneHotEncoder(drop='first', sparse=False)
encoded_features = encoder.fit_transform(df[categorical_columns])

# Create a new DataFrame with the encoded features and the numeric columns
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_columns))
data_processed = pd.concat([encoded_df, df[numeric_columns]], axis=1)
```

In [25]:
```python
# Step 2: Feature Engineering - Create interaction terms and standardize numeric columns
# For simplicity, I will create an interaction term between 'job_title_ML Ops' and 'experience_level_Senior'
data_processed['job_title_ML Ops * experience_level_Senior'] = data_processed['job_title_ML Ops'] * data_processed['experience_level_Senior']

# Standardize numeric columns
scaler = StandardScaler()
data_processed[numeric_columns] = scaler.fit_transform(data_processed[numeric_columns])
```

In [26]:
```python
# Step 3: Split the data into training and testing sets
X = data_processed.drop('salary', axis=1)
y = data_processed['salary']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [27]:
```python
# Step 4: Build the Multiple Linear Regression model with Ridge regularization
ridge_regression_model = Ridge(alpha=1.0)  # Use alpha value as needed for regularization strength
ridge_regression_model.fit(X_train, y_train)
```

Out[27]:
```
Ridge()
```

In [28]:
```python
# Step 5: Build the Decision Tree Regression model
decision_tree_model = DecisionTreeRegressor(random_state=42)
decision_tree_model.fit(X_train, y_train)
```

Out[28]:
```
DecisionTreeRegressor(random_state=42)
```

In [29]:
```python
# Step 6: Evaluate the models
# For the Ridge Regression model
y_pred_ridge = ridge_regression_model.predict(X_test)
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
print(f"Ridge Regression Mean Squared Error: {mse_ridge}")

# For the Decision Tree Regression model
y_pred_tree = decision_tree_model.predict(X_test)
mse_tree = mean_squared_error(y_test, y_pred_tree)
print(f"Decision Tree Regression Mean Squared Error: {mse_tree}")
```

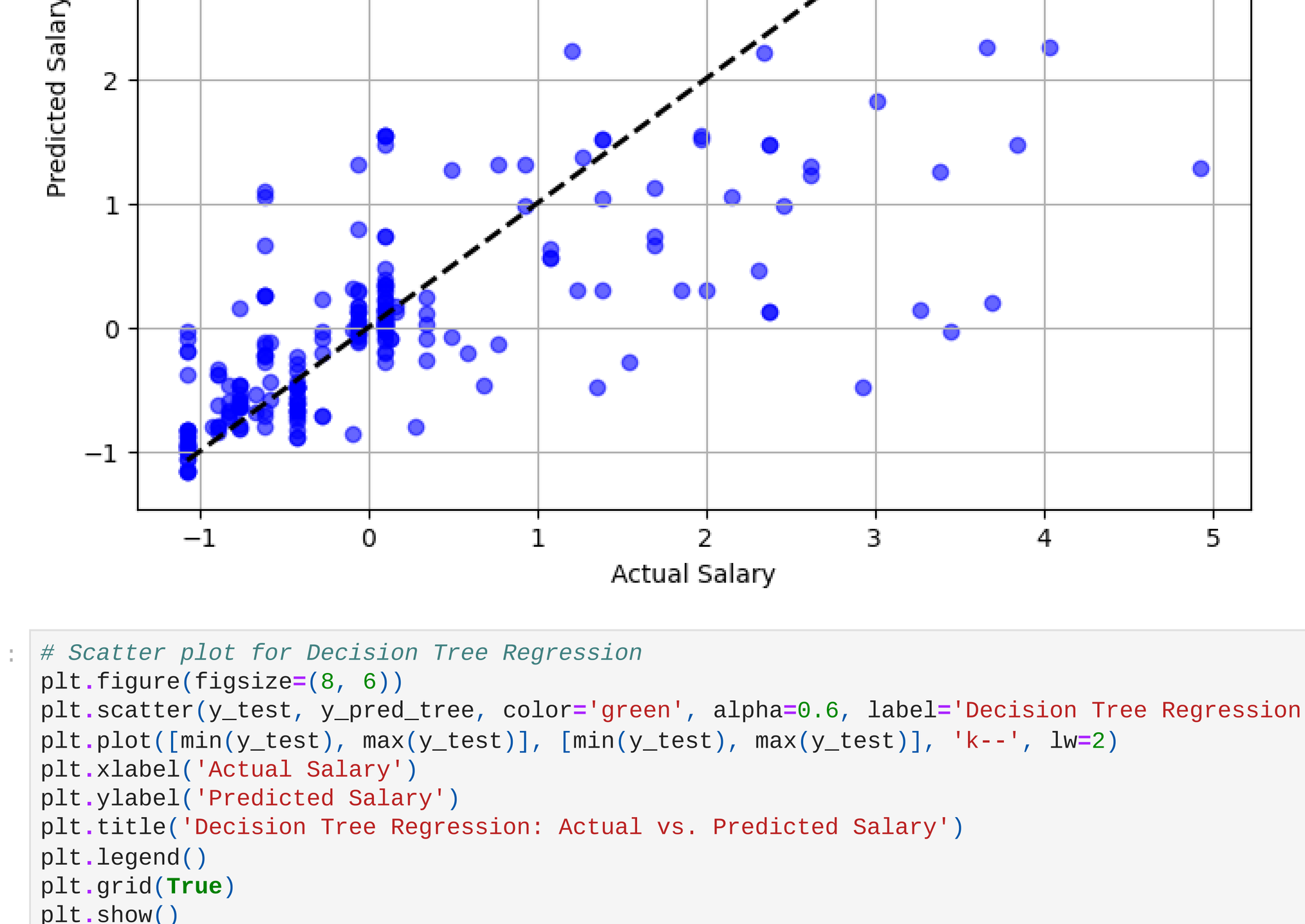
```
Ridge Regression Mean Squared Error: 0.6658440983580333
Decision Tree Regression Mean Squared Error: 0.8698009723614151
```

The MSE for the Ridge Regression model is approximately 0.67. Since the MSE is relatively low, it suggests that the Ridge Regression model is making accurate predictions, and the predicted salary values are quite close to the actual salary values in the dataset.

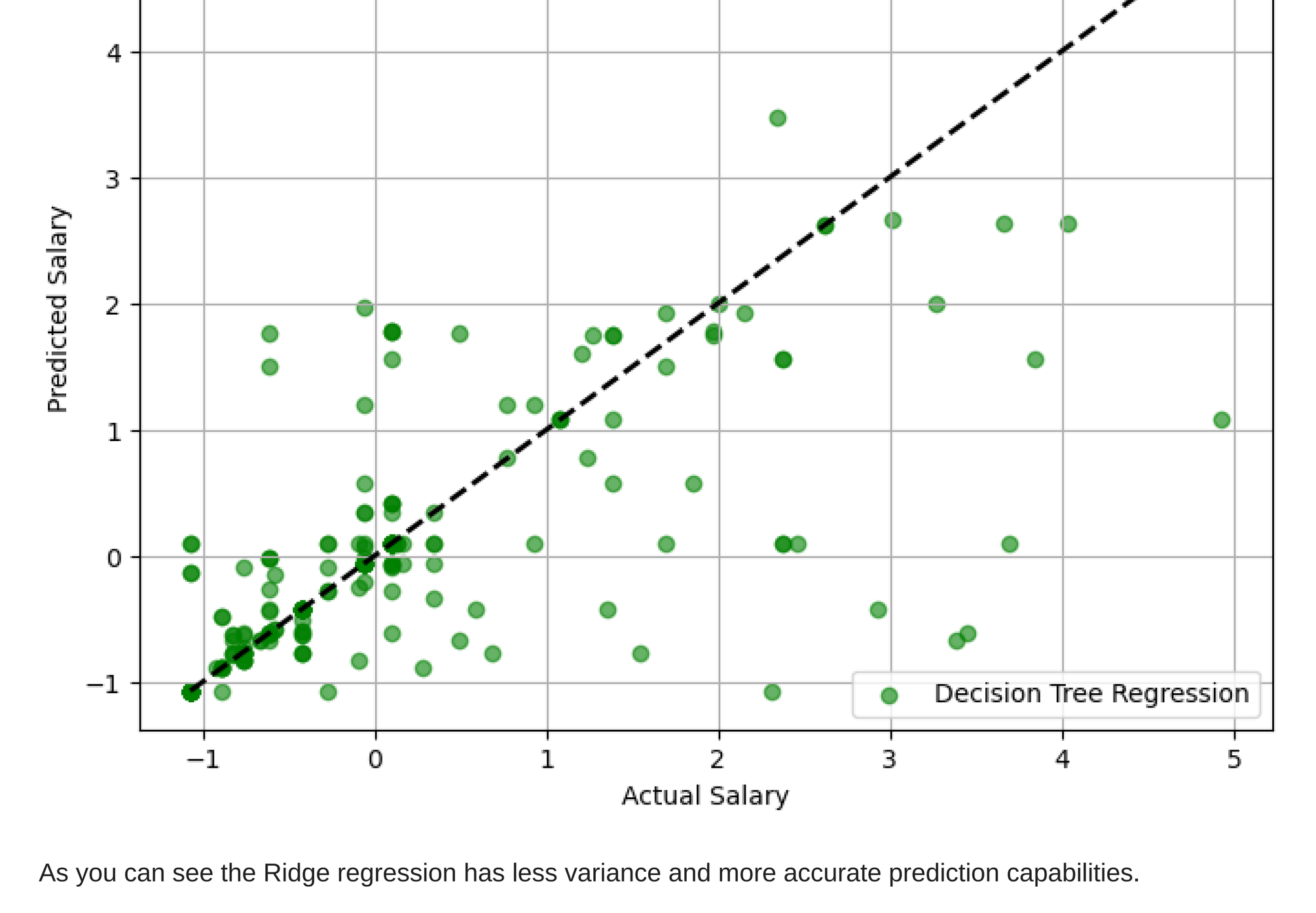
The MSE for the Decision Tree Regression model is approximately 0.87. Similar to the Ridge Regression model, the Decision Tree model also has a relatively low MSE, indicating that it is making accurate predictions, and the predicted salary values are close to the actual salary values in the dataset.

A lower MSE implies better predictive performance, as it means the model's predicted salary values are closer to the actual salary values. Therefore, based on the MSE values, the Ridge Regression model is performing slightly better in predicting the salaries compared to the Decision Tree Regression model. This model with the location included is overall better in comparison with the one without location included.

In [30]:
```python
# Scatter plot for Ridge Regression
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_ridge, color='blue', alpha=0.6, label='Ridge Regression')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'k--', lw=2)
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.title('Ridge Regression: Actual vs. Predicted Salary')
plt.legend()
plt.grid(True)
plt.show()
```



In [31]:
```python
# Scatter plot for Decision Tree Regression
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_tree, color='green', alpha=0.6, label='Decision Tree Regression')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'k--', lw=2)
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.title('Decision Tree Regression: Actual vs. Predicted Salary')
plt.legend()
plt.grid(True)
plt.show()
```



As you can see the Ridge regression has less variance and more accurate prediction capabilities.

In [ ]: