Data Mining Term Project

Import necessary libraries

[3]:
```
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import warnings
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import SVR
from sklearn.linear_model import BayesianRidge
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report,
 confusion_matrix
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
warnings.filterwarnings('ignore')
```

Read the file

[4]:
```
df = pd.read_csv('ds_salaries.csv', encoding='utf-8')
```

Calculate the number of rows to delete (30% of the total rows)

[5]:
```
rows_to_delete = int(0.3 * len(df))
```

Randomly select rows to delete

[6]:
```
rows_indices_to_delete = np.random.choice(df.index, size=rows_to_delete,
 replace=False)
```

Mark the selected rows as missing or set to a specific value

[7]:
```
df.loc[rows_indices_to_delete, 'salary'] = np.nan
df.loc[rows_indices_to_delete, 'salary_in_usd'] = np.nan
```

[8]:
```
df.reset_index(drop=True, inplace=True)
```

drop duplicate rows

[9]:
```
duplicate_rows = df[df.duplicated()]
df_no_duplicates = df.drop_duplicates()
```

First look on data

[10]:
```
df.head()
```

[10]:
```
   work_year experience_level employment_type                job_title  \
0       2023               SE              FT  Principal Data Scientist
1       2023               MI              CT              ML Engineer
2       2023               MI              CT              ML Engineer
3       2023               SE              FT           Data Scientist
4       2023               SE              FT           Data Scientist

     salary salary_currency  salary_in_usd employee_residence  remote_ratio  \
0   80000.0             EUR        85847.0                 ES           100
1   30000.0             USD        30000.0                 US           100
2   25500.0             USD        25500.0                 US           100
3  175000.0             USD       175000.0                 CA           100
4       NaN             USD            NaN                 CA           100

  company_location company_size
0               ES            L
1               US            S
2               US            S
3               CA            M
4               CA            M
```

[11]:
```
df.tail()
```

[11]:
```
      work_year experience_level employment_type                 job_title  \
3750       2020               SE              FT            Data Scientist
3751       2021               MI              FT  Principal Data Scientist
3752       2020               EN              FT            Data Scientist
3753       2020               EN              CT     Business Data Analyst
3754       2021               SE              FT      Data Science Manager

      salary salary_currency  salary_in_usd employee_residence  \
3750     NaN             USD            NaN                 US
```

```
3751   151000.0        USD    151000.0             US
3752   105000.0        USD    105000.0             US
3753   100000.0        USD    100000.0             US
3754  7000000.0        INR     94665.0             IN

      remote_ratio company_location company_size
3750           100               US            L
3751           100               US            L
3752           100               US            S
3753           100               US            L
3754            50               IN            L
```

droping missing values

[12] :
```
df.dropna(inplace=True)
df_cleaned = df.dropna()
```

Splitting Numerical from Categorical features

[13] :
```
numerical_features = list(set(df.columns.to_list()) -
 {'salary','salary_in_usd','remote_ratio','work_year'})
categorical_features = list(set(df.columns.to_list()) - set(numerical_features))
```

select which column to normalize

[14] :
```
column_to_normalize = 'salary_in_usd'
scaler = MinMaxScaler()
df[column_to_normalize] = scaler.fit_transform(df[[column_to_normalize]])
```

discretization of salary_in_usd

[15] :
```
column_to_discretize = 'salary_in_usd'
bin_edges = [0.000622657,0.306592,0.914581,1]
bin_labels = ['LOW', 'MEDIUM', 'HIGH']
df['discretized_' + column_to_discretize] = pd.cut(df[column_to_discretize],
 bins=bin_edges, labels=bin_labels)
le = LabelEncoder()
df['discretized_salary_in_usd'] = le.
 fit_transform(df['discretized_salary_in_usd'])
```

desribe the median of salary_in_usd attribut

[16] :
```
attribute_column = 'salary_in_usd'
median_value = df[attribute_column].median()
print(f"The median value of the '{attribute_column}' column is: {median_value}")
```

```
The median value of the 'salary_in_usd' column is: 0.3049725832775606
```

Filtering rows

[17] :
```
df = df[df['salary_in_usd'] > 0]

#filtering and keeping only rows where 'discretized_salary_in_usd' is not nan
df = df[df['discretized_salary_in_usd'] != np.nan]
```

Print the modified dataset

[18] :
```
print(df)
print(df.isnull().sum())
print(df.describe())
```

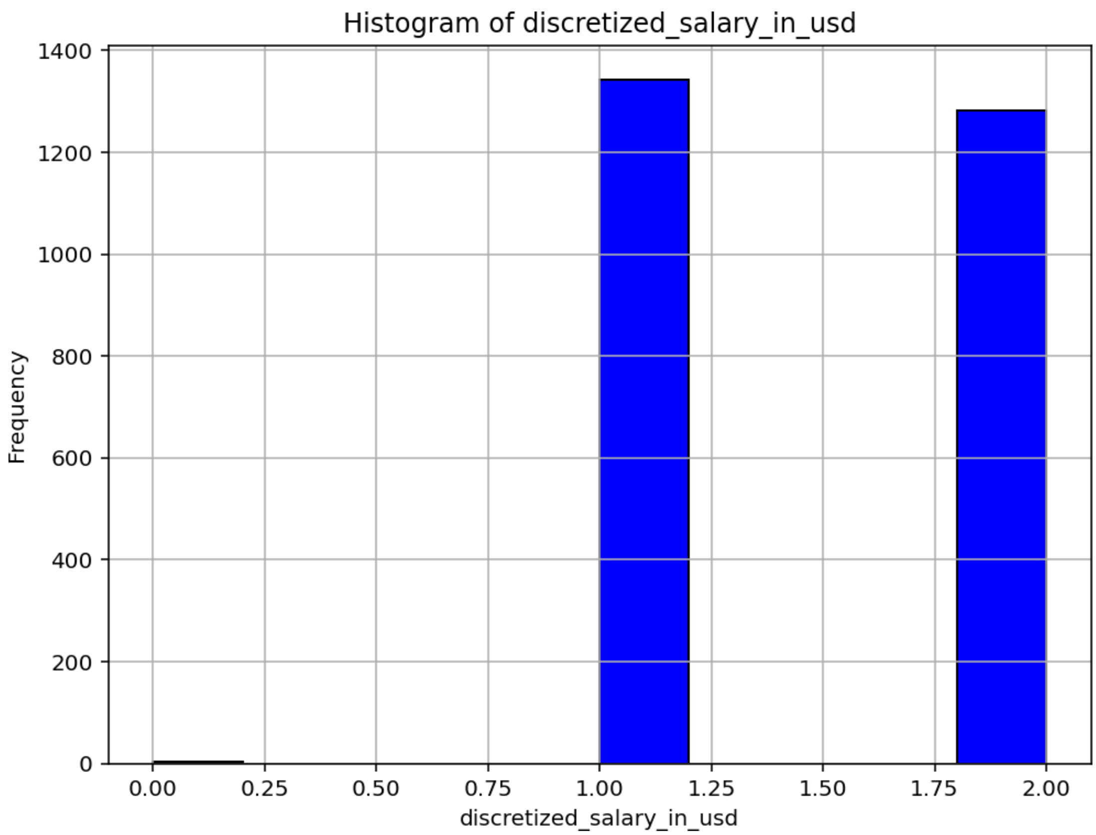
```
      work_year experience_level employment_type               job_title  \
0          2023               SE              FT  Principal Data Scientist
1          2023               MI              CT             ML Engineer
2          2023               MI              CT             ML Engineer
3          2023               SE              FT          Data Scientist
6          2023               SE              FT       Applied Scientist
...         ...              ...             ...                     ...
3749       2021               SE              FT         Data Specialist
3751       2021               MI              FT  Principal Data Scientist
3752       2020               EN              FT          Data Scientist
3753       2020               EN              CT   Business Data Analyst
3754       2021               SE              FT    Data Science Manager

         salary salary_currency  salary_in_usd employee_residence  \
0       80000.0             EUR       0.189545                 ES
1       30000.0             USD       0.058398                 US
2       25500.0             USD       0.047831                 US
3      175000.0             USD       0.398906                 CA
6      136000.0             USD       0.307321                 US
...         ...             ...            ...                ...
3749   165000.0             USD       0.375422                 US
3751   151000.0             USD       0.342546                 US
3752   105000.0             USD       0.234523                 US
3753   100000.0             USD       0.222781                 US
3754  7000000.0             INR       0.210253                 IN

      remote_ratio company_location company_size  discretized_salary_in_usd
0              100               ES            L                          1
1              100               US            S                          1
2              100               US            S                          1
3              100               CA            M                          2
6                0               US            L                          2
...            ...              ...          ...                        ...
3749           100               US            L                          2
3751           100               US            L                          2
3752           100               US            S                          1
3753           100               US            L                          1
```

```
3754         50         IN       L              1

[2628 rows x 12 columns]
work_year                  0
experience_level           0
employment_type            0
job_title                  0
salary                     0
salary_currency            0
salary_in_usd              0
employee_residence         0
remote_ratio               0
company_location           0
company_size               0
discretized_salary_in_usd  0
dtype: int64
         work_year        salary  salary_in_usd  remote_ratio  \
count  2628.000000  2.628000e+03    2628.000000   2628.000000
mean   2022.374429  1.955852e+05       0.311638     45.985540
std       0.690785  7.488385e+05       0.146846     48.581506
min    2020.000000  6.000000e+03       0.000650      0.000000
25%    2022.000000  1.000000e+05       0.211039      0.000000
50%    2022.000000  1.387500e+05       0.304973      0.000000
75%    2023.000000  1.800000e+05       0.398906    100.000000
max    2023.000000  3.040000e+07       1.000000    100.000000

       discretized_salary_in_usd
count                2628.000000
mean                    1.486301
std                     0.502944
min                     0.000000
25%                     1.000000
50%                     1.000000
75%                     2.000000
max                     2.000000
```

Generate a histogram

[19]:
```
column_of_interest = 'discretized_salary_in_usd'
plt.figure(figsize=(8, 6))
plt.hist(df[column_of_interest], bins=10, color='blue', edgecolor='black')
plt.title(f'Histogram of {column_of_interest}')
plt.xlabel(column_of_interest)
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

[19]:



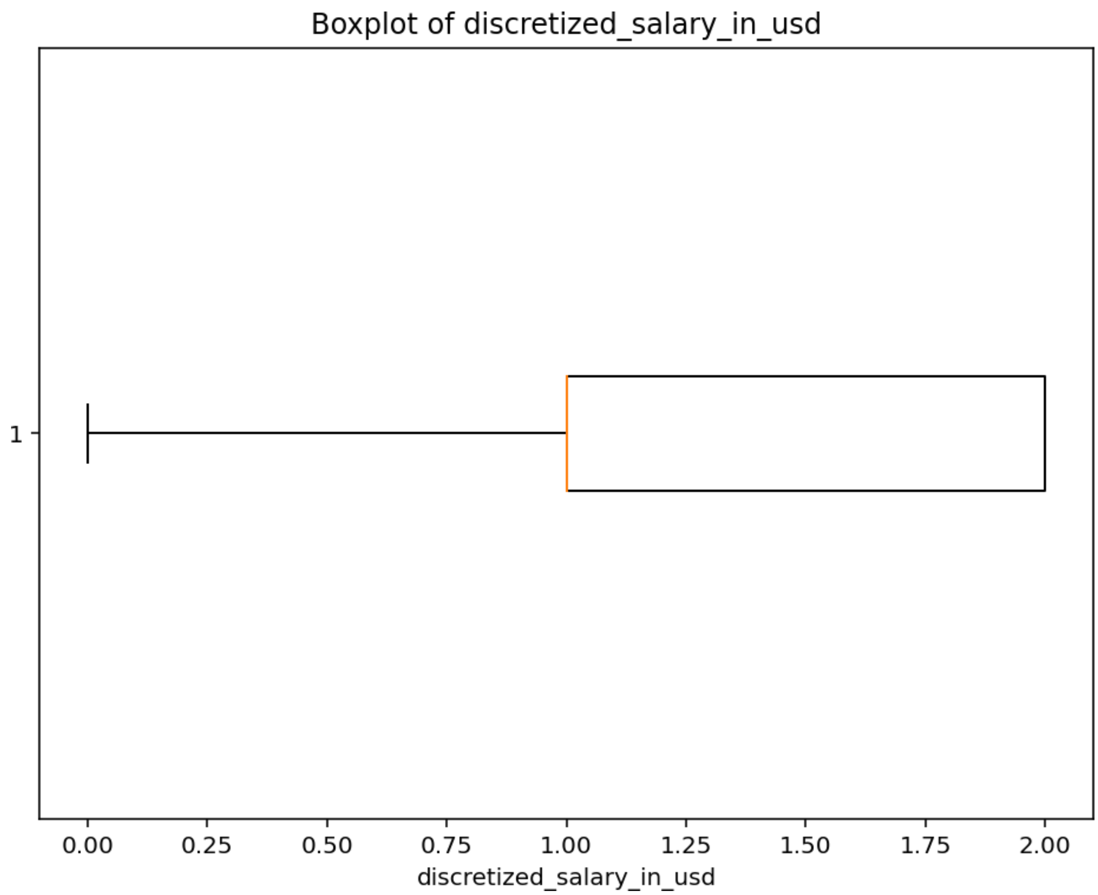
This distribution suggests that the majority of individuals fall within these two distinct salary ranges, with a higher concentration in the lower range.

Generate a boxplot

[20] :
```
plt.figure(figsize=(8, 6))
plt.boxplot(df[column_of_interest],  vert=False)
plt.title(f'Boxplot of {column_of_interest}')
plt.xlabel(column_of_interest)
```

[20] : Text(0.5, 0, 'discretized_salary_in_usd')

[20]:

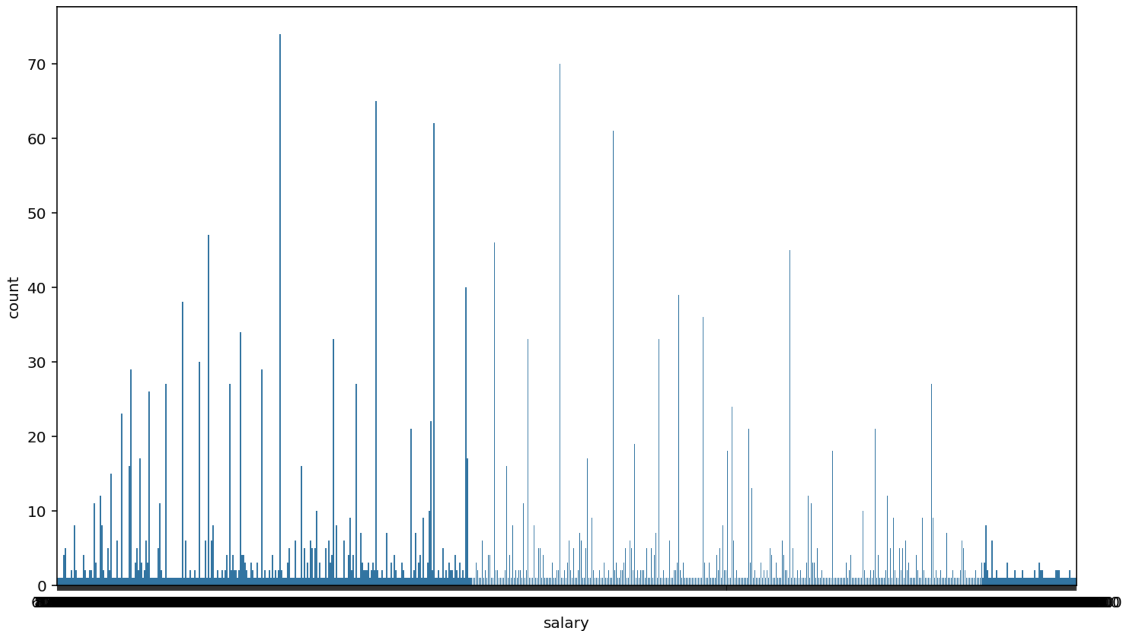


The clean boxplot, combined with the absence of outliers, underscores a consistent salary distribution without significant deviations from the central tendency

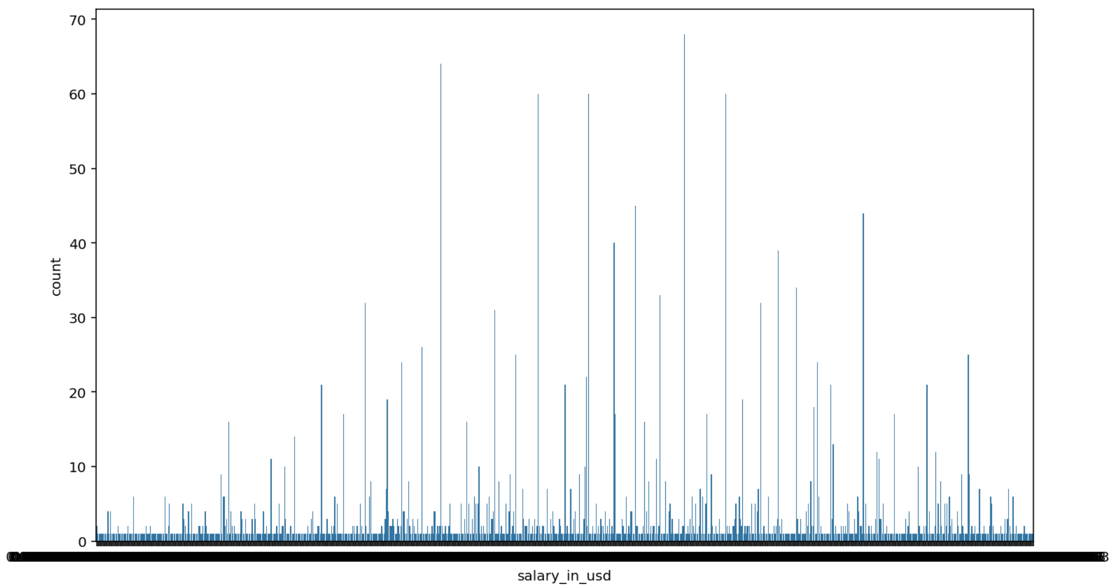
Create countplots for numerical attributes

[21] :
```
sns.countplot(x=df['salary'])
plt.show()
sns.countplot(x=df['salary_in_usd'])
plt.show()
sns.countplot(x=df['remote_ratio'])
plt.show()
sns.countplot(x=df['work_year'])
plt.show()
```

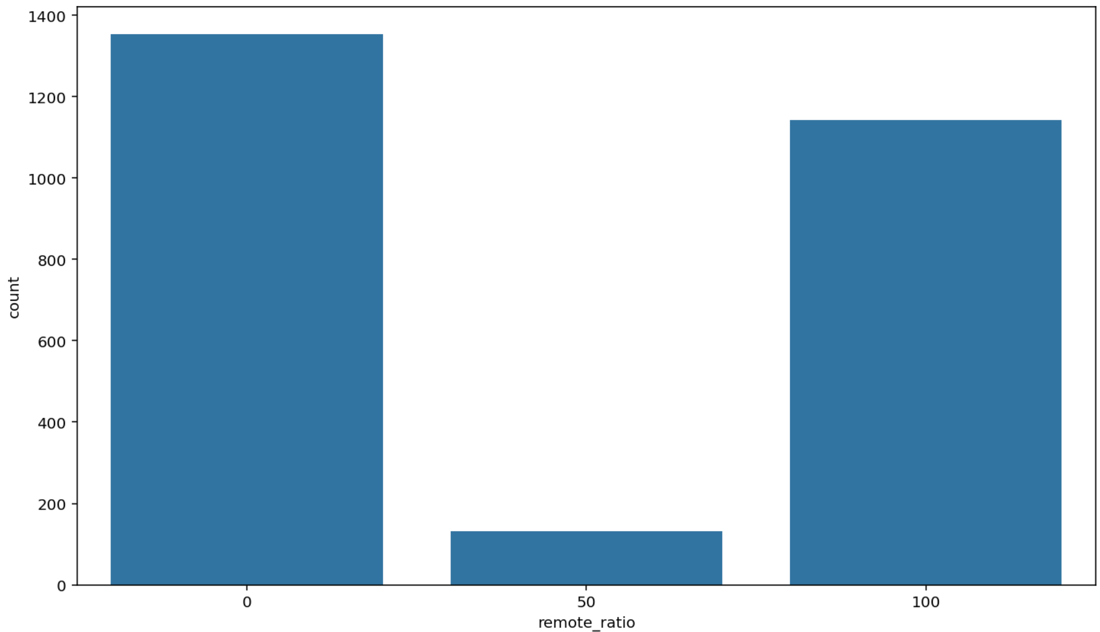
[21]:



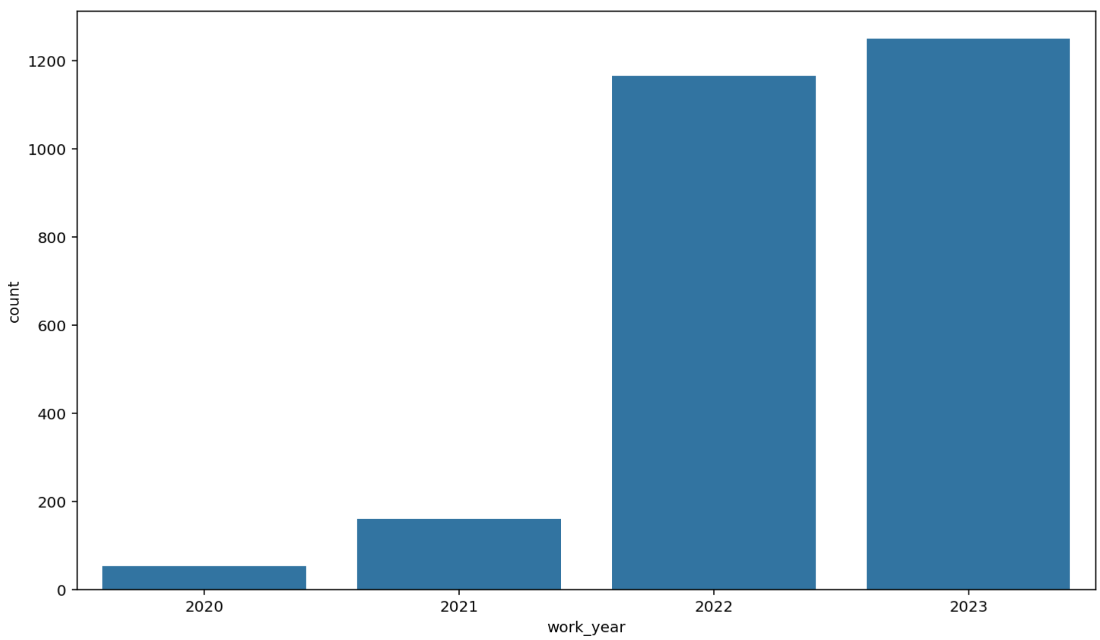
[21]:



[21]:



[21]:



The salary data analysis reveals variability in compensation, attributed to factors such as experience levels and job titles. Visualization of salaries in USD illustrates that only a few employees receive the highest incomes, likely linked to their extensive experience. Remote work ratios are predominantly concentrated at 0%, 50%, and 100%, with most employees either fully remote or not at all. Analysis of work years indicates a surge in new hires in 2022 and 2023, suggesting a growing workforce, while fewer employees are recorded from 2020. This trend suggests an influx of new hires, particularly in the recent years of 2022 and 2023, contributing to the overall increase in the workforce.

0.0.1 Regression

In this section, the code performs Bayesian Ridge Regression on the dataset, predicts salaries based on company size, and evaluates the model's performance .

[22]:
```
le = LabelEncoder()

 # create a copy of the dataframe
df_copy = df.copy()
df_copy['company_size'] = le.fit_transform(df_copy['company_size'])
print(le.classes_)

 # Drop columns that are not needed for regression
df_copy.drop(columns=['work_year', 'experience_level', 'employment_type',␣
 ↪'job_title', 'salary_currency',
                                    'employee_residence', 'company_location'],␣
 ↪inplace=True)

 # Assuming 'company_size' is a feature and 'salary_in_usd' is the target␣
 ↪variable
X = df_copy[['company_size']]
y = df_copy['salary_in_usd']

 # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

 # Fit Bayesian Ridge Regression
regressor = BayesianRidge()
regressor.fit(X_train, y_train)

 # Make predictions on the test set
y_pred = regressor.predict(X_test)

 # Calculate and print the Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

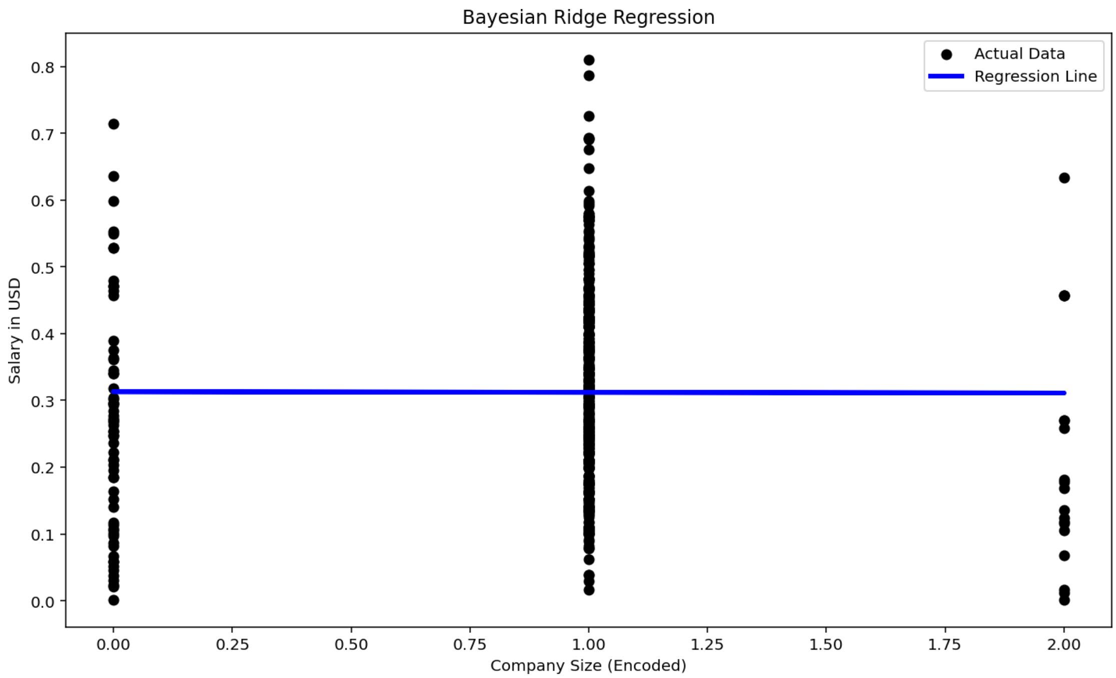
```
['L' 'M' 'S']
Mean Squared Error: 0.020944256795436173
```

Visualize the regression line

[23]:
```
plt.scatter(X_test, y_test, color='black', label='Actual Data')
plt.plot(X_test, y_pred, color='blue', linewidth=3, label='Regression Line')
```

```
plt.title('Bayesian Ridge Regression')
plt.xlabel('Company Size (Encoded)')
plt.ylabel('Salary in USD')
plt.legend()
plt.show()
```

[23]:



In this baysian ridge regression we compare the company size with the salary in usd and as we can understand from the graph that we have the regression line in the 150000 and also from our data we have three company size the Land the S and the M and eaxh of the mis representing here as 0 and 1and 2 and we can see that most of the data is in the M which is 1 and are above the regression line which is nearly the average of the data or of the salary comparing with the company size and we have a poor number in term of the L size in the company which we don't have much employee here comparing to the other sizes .

Classification Assuming 'job_title' is a feature and 'employment_type' is the target variable and also In this part we can seperate it into part to make it more detailled so we have :

[24] :

```
df_copy = df.copy()

# Drop columns that are not needed for classification
df_copy.drop(columns=['work_year', 'experience_level', 'salary_in_usd',␣
↪'salary_currency',
                     'employee_residence', 'company_location',␣
↪'company_size'], inplace=True)
```

The next part creates a copy of the original DataFrame (df) and drops columns that are not needed for text classification, leaving only the relevant features and target variable.

[25] :
```
X = df_copy['job_title']
y = df_copy['employment_type']
```

Split the data into training and testing sets,Use CountVectorizer to convert job titles into numerical features

[26] :
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 ↪random_state=42)

# Use CountVectorizer to convert job titles into numerical features
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

[27] :
```
# Train a Multinomial Naive Bayes classifier
naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(X_train_vectorized, y_train)

# Make predictions on the test set
y_pred = naive_bayes_classifier.predict(X_test_vectorized)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

[28] :
```
print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{class_report}')
cm = confusion_matrix(y_test, y_pred)
```

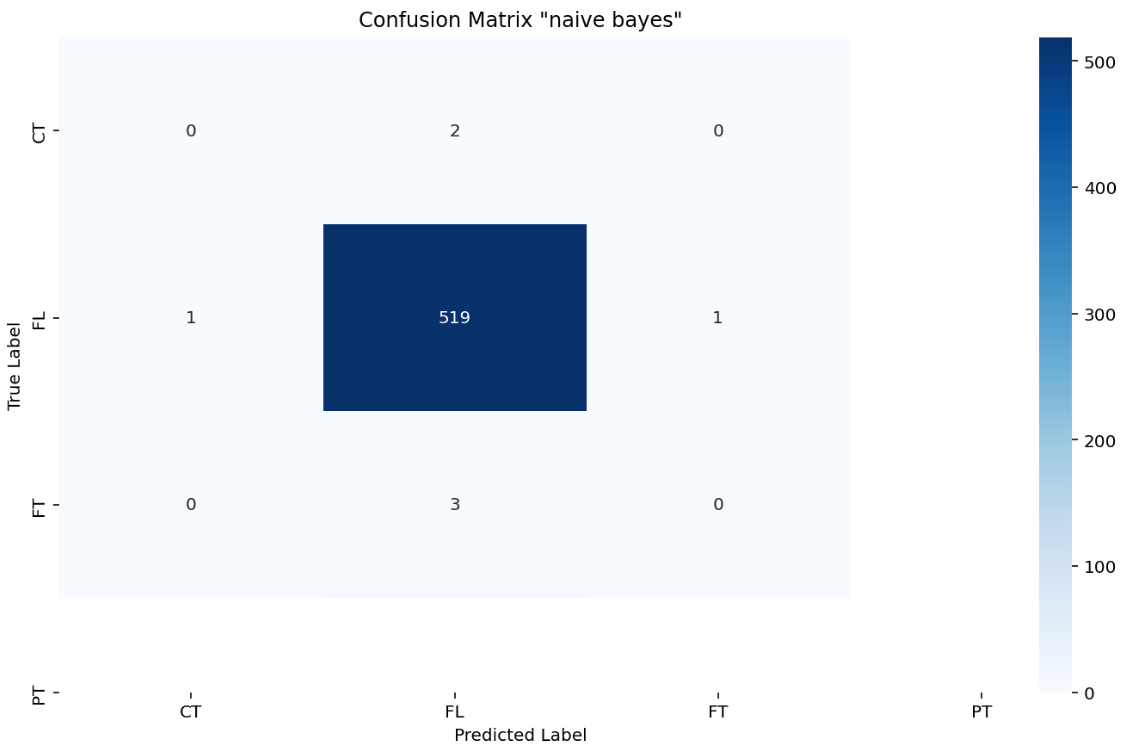
```
Accuracy: 0.9866920152091255
Confusion Matrix:
[[  0   2   0]
 [  1 519   1]
 [  0   3   0]]
Classification Report:
              precision    recall  f1-score   support

          FL       0.00      0.00      0.00         2
          FT       0.99      1.00      0.99       521
          PT       0.00      0.00      0.00         3

    accuracy                           0.99       526
   macro avg       0.33      0.33      0.33       526
```

```
weighted avg       0.98      0.99      0.98       526
```

Create a heatmap for the confusion matrix This part creates a heatmap of the confusion matrix using Seaborn. The heatmap visually represents the true and predicted labels, making it easier to interpret the performance of the Naive Bayes classifier.

[29] :
```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
  xticklabels=naive_bayes_classifier.classes_,
  yticklabels=naive_bayes_classifier.classes_)
plt.title('Confusion Matrix "naive bayes"')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

[29]:



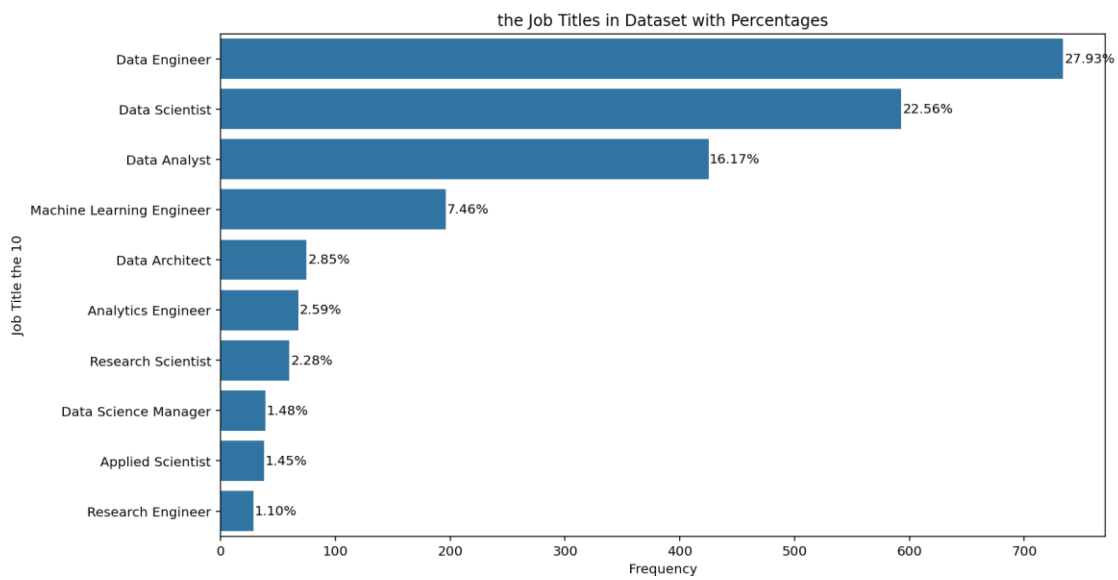
And from the confusion matrix we can see that we have the counter of the label is higher in the FL which we are talking here about the employement type which is the less number of the employement .

Calculate percentage visualizations help in understanding the distribution of job titles and experience levels in the dataset

[30] :

```
job_count = df['job_title'].value_counts().nlargest(10)
total_jobs = len(df['job_title'])
percentages = (job_count / total_jobs) * 100

# Create a horizontal bar plot
ax = sns.barplot(x=job_count.values, y=job_count.index, orient='h')

# Annotate bars with percentages
for i, v in enumerate(job_count.values):
    percentage = percentages.iloc[i]
    ax.text(v + 1, i, f'{percentage:.2f}%', va='center', fontsize=10)

# Customize plot labels and title
plt.title('the Job Titles in Dataset with Percentages')
plt.xlabel('Frequency')
plt.ylabel('Job Title the 10')
# Display the plot
plt.show()
```

[30]:



And we can understand from here that the higher percentage in the job titles is the data engineering in the company the fewer one is the research engineering which have a lower frequency and percentage.

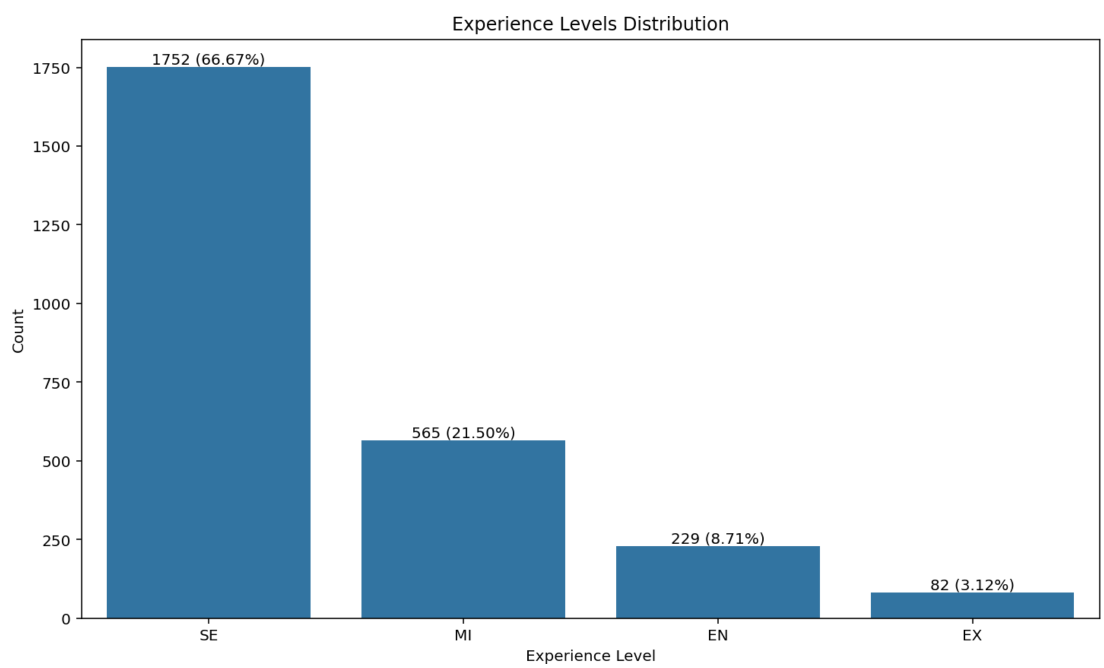
Create a count plot for experience levels

[31] :

```
ax = sns.countplot(x=df['experience_level'])

# Annotate bars with counts or percentages
total_records = len(df['experience_level'])
```

```
counts = df['experience_level'].value_counts()

# Calculate percentages
percentages = (counts / total_records) * 100

# Annotate bars with percentages or counts
for i, v in enumerate(counts):
    percentage = percentages.iloc[i]
    ax.text(i, v + 1, f'{v} ({percentage:.2f}%)', ha='center', va='bottom',␣
 ↪fontsize=10)

# Customize plot labels and title
plt.title('Experience Levels Distribution')
plt.xlabel('Experience Level')
plt.ylabel('Count')
plt.show()
```

[31]:



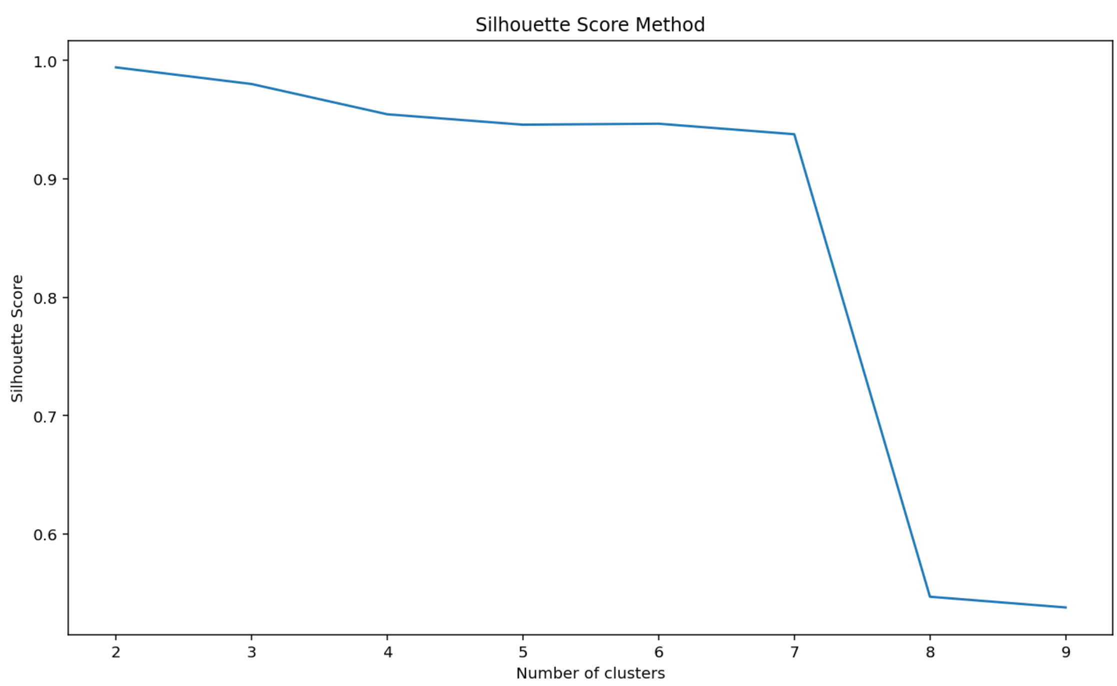
We have here in the experience levels 4 levels which are the SE (Software Engineer) and MI (Mid-level) and EN (Entry-level) and EX (Executive or Experienced) and from the stats chart we have that the higher percentage goes to software engineering « MI » with 67 percentage and the expert are very low .

Clustering

[32] :
```
le = LabelEncoder()

# create a copy of the dataframe
df_copy = df.copy()
df_copy['company_size'] = le.fit_transform(df_copy['company_size'])
print(le.classes_)

# Drop columns that are not needed for clustering
df_copy.drop(columns=['work_year', 'experience_level', 'employment_type',␣
↪'job_title', 'salary_in_usd', 'salary_currency',
                    'employee_residence', 'company_location'],␣
↪inplace=True)

silhouette_scores = []
for k in range(2, 10):  # Start from 2 clusters as silhouette score requires at␣
↪least 2 clusters
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df_copy)
    score = silhouette_score(df_copy, kmeans.labels_)
    silhouette_scores.append(score)

plt.plot(range(2, 10), silhouette_scores)
plt.title('Silhouette Score Method')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.show()
```

```
['L' 'M' 'S']
```

[32]:

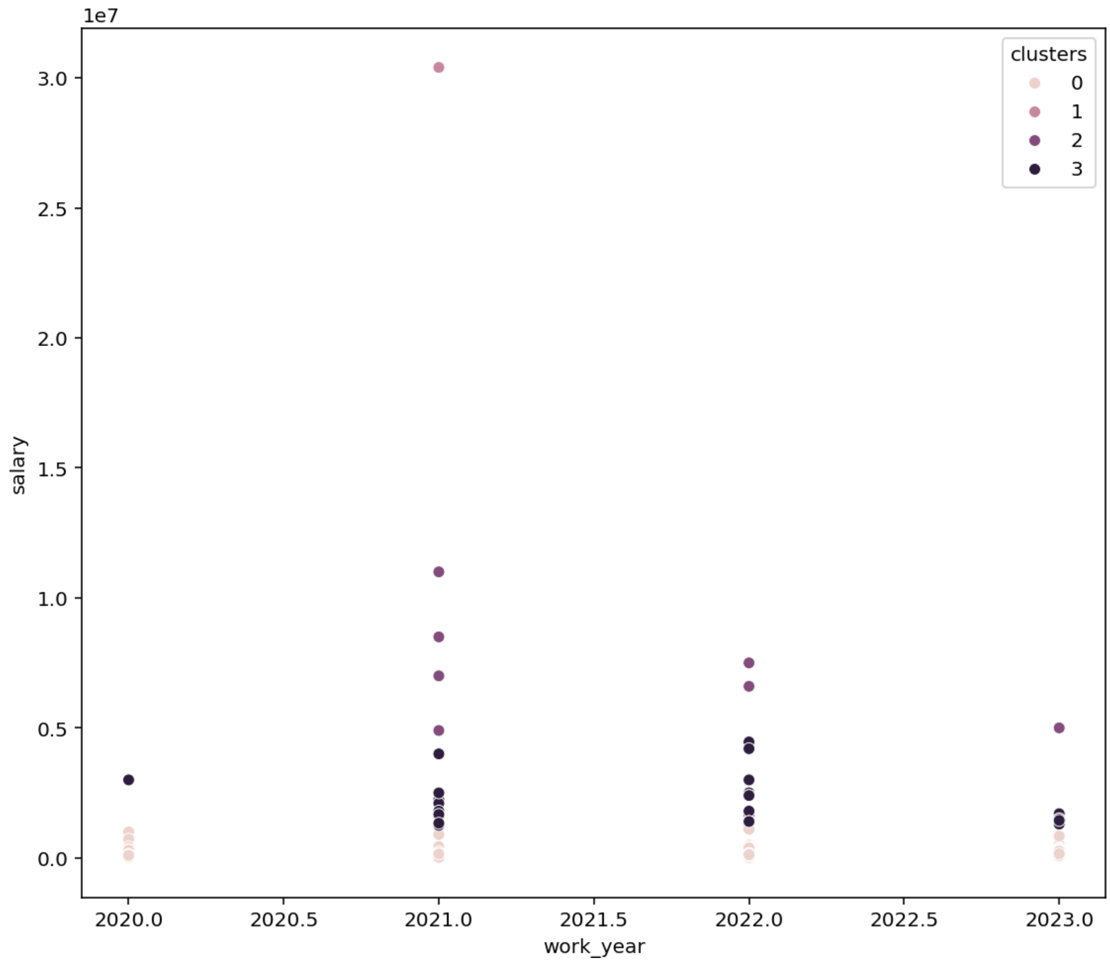


We can understand from the silhouette score chart that the score is keep decreasing comparing to the number of the clusters silhouette score started decreasing which means that the clusters are assigned in the wrong way. Since we know the higher the silhouette score, the chances get higher to the optimal one, so it can be said that the probable number of clusters should be 10 or more.

Kmeans cluster

[33] :
```
kmeans = KMeans(n_clusters=4)
kmeans.fit(df_copy)

df['clusters'] = kmeans.predict(df_copy)
sns.scatterplot(df, x='work_year', y='salary', hue='clusters')
plt.gcf().set_size_inches(8,7)
plt.tight_layout()
plt.show();
```

[33]:



And in the final chart of the clustering we have this clusters comparing the salary with the work years which we can see clearly that the data that we have is regrouping in 2020 and 2021 and 2022 and 2023 and the highest one are in 2021 and when we examine the average values of ‘work_year’ and ‘salary’ for each cluster we see that we have a difference in clustering

Project collaborators

Wail Zita : 21091000199

Yasmen yasser mohamed abdeltawwab : 23352525056

Khadija Brik : 21091010094

Hiba EL Quassimi : 21091010095