King Saud University

College of Computer and Information Sciences

Information Technology department

IT 326: Data Mining

Course Project

AI Employment Trends Analysis

Project Final Report

Group#:	**8**	
Section#:	**71162**	
Group Members	**Name**	**ID**
	Rawan albatati	**443204566**
	Jood Alkhrashi	**444203007**
	Najla Almazyad	**444200948**
	Ghadi Alzahrani	**443200533**
	Maha Aljandoul	**444200646**

1. Problem

Artificial Intelligence (AI) and automation are fundamentally altering the job market, creating both opportunities and challenges for various industries. The widespread adoption of these technologies raises critical questions about their impact on employment, salaries, and required skills. This project seeks to analyze how AI adoption and automation risks are distributed across different job roles and industries, helping to identify the areas most impacted by these advancements, understanding these trends is crucial for preparing the workforce for the future. By examining factors such as AI adoption levels, job growth projections, and salary ranges, this study aims to provide actionable insights for policymakers, educators, and job seekers, enabling them to adapt to a rapidly changing job market.

2. Data Mining Task

In our project, we employ two data mining tasks to analyze the impact of AI adoption on the job market: classification and clustering.

For classification, we train our model to predict the level of AI adoption in job roles, categorizing them into three classes: low, medium, and high. This task uses features such as company size, salary, and required skills to determine the class label. The goal of this task is to accurately identify the adoption levels for different roles, enabling insights into which industries and jobs are leading in AI integration.

As for clustering, our model groups job roles based on shared characteristics without considering predefined labels such as Job Growth Projection. The clustering process focuses on attributes like salary, remote work compatibility, and automation risks. The goal is to identify meaningful patterns and trends in the data, providing insights into which roles are more resilient or vulnerable to automation.

3. Data

The source: (https://www.kaggle.com/datasets/uom190346a/ai-powered-job-market-insights)

Number of attributes: 10

Number of Objects: 500

Class label: Job_Growth_Projection

Attributes Description:

```
import pandas as pd

file_path = 'ai_job_market_insights.csv'
df = pd.read_csv(file_path)

num_attributes = df.shape[1]

attribute_types = df.dtypes

num_objects = df.shape[0]

class_label_name = df.columns[-1]

print(f"Number of attributes: {num_attributes}")
print(f"Type of attributes:\n{attribute_types}")
print(f"Number of objects: {num_objects}")
print(f"Class or label name: {class_label_name}")

print("First 5 lines:")
print(df.head())

print("\nLast 5 lines:")
print(df.tail())
```

Attribute Name	Description	Data Type	Possible Values
Job_Title	Represents the title of the job, such as 'Data Scientist' or 'Software Engineer.'	Categorical	Strings (e.g., 'Data Scientist', 'Engineer')
Industry	Represents the industry type where the job belongs, such as 'Technology' or 'Healthcare.'	Categorical	Strings (e.g., 'Technology', 'Healthcare')
Company_Size	Indicates the size of the company as small, medium, or large.	Categorical	'Small', 'Medium', 'Large'
Remote_Friendly	Indicates if the job allows remote work.	Binary	1: 'Yes', 0: 'No'
Salary_USD	Represents the salary in USD for the job role.	Numeric	Continuous values in USD
AI_Adoption_Level	Indicates the level of AI adoption in the job role.	Numeric	1: 'Low', 2: 'Medium', 3: 'High'
Automation_Risk	Represents the likelihood of automation for the job role.	Numeric	Continuous values (e.g., 0.0 - 1.0)
Required_Skills	Represents the skills required for the job role.	Categorical	Strings (e.g., 'Python, Machine Learning')
Location	Represents the job's geographical location.	Categorical	Strings (e.g., 'USA', 'UK', 'Remote')
Job_Growth_Projection	Predicts the future demand for the job role over the next few years.	Numeric	Continuous values (e.g., -10% to +50%)

Missing values

```
#-------------------------

# 3. Check for missing values
missing_values = df.isnull().sum()
print("\nMissing values in each column:")
print(missing_values)

# 4. Statistical Summary (Mean, Variance, etc.)
print("\nStatistical Summary:")
print(df.describe())

print("\nThe variance:")
var_data=df["Salary_USD"].var()
print(var_data)
```

```
Missing values in each column:
Job_Title                0
Industry                 0
Company_Size             0
Location                 0
AI_Adoption_Level        0
Automation_Risk          0
Required_Skills          0
Salary_USD               0
Remote_Friendly          0
Job_Growth_Projection    0
dtype: int64
```

There are no missing values in the dataset, as indicated by the zero counts in each column.

Statical Measures for each numeric column:

Show Five Number Summary

```
print(df.describe())
```

```
Statistical Summary:
          Salary_USD
count     500.000000
mean    91222.390974
std     20504.291453
min     31969.526346
25%     78511.514863
50%     91998.195286
75%    103971.282092
max    155209.821614
```

The Salary_USD attribute shows significant variability, ranging from $31,969.53 to $155,209.82, with a mean of $91,222.39 and a median of $91,998.20, indicating a relatively balanced distribution. The 1st quartile (Q1) is $78,511.51, and the 3rd quartile (Q3) is $103,971.28, highlighting the middle 50% of salaries. However, the standard deviation of $20,504.29 and the presence of outliers (salaries below $35,000 and above $140,000) suggest the need for preprocessing to handle extreme values and improve data consistency.

Show the Variance:

```
print("\nThe variance:")
var_data=df["Salary_USD"].var()
print(var_data)
```

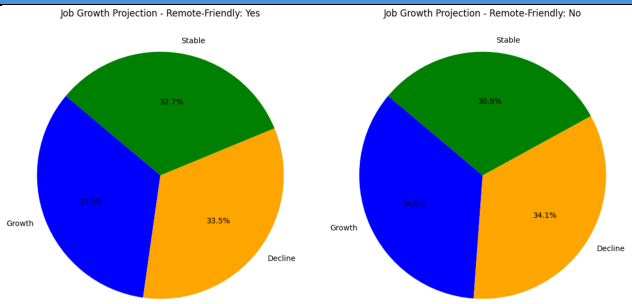
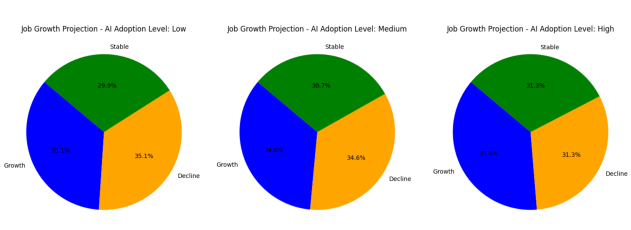
```
The variance:
420425968.00916165
```

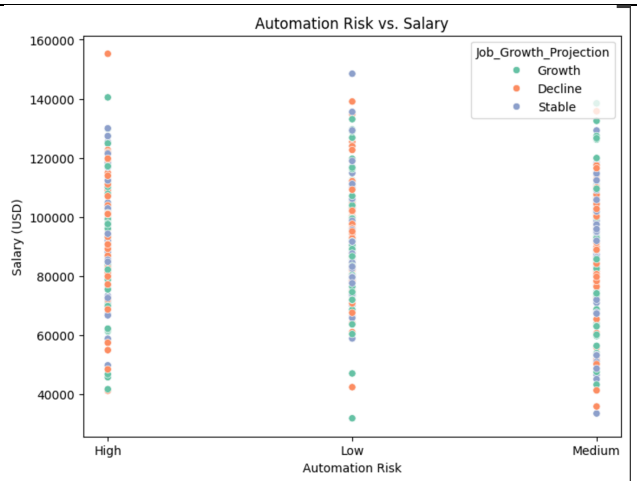
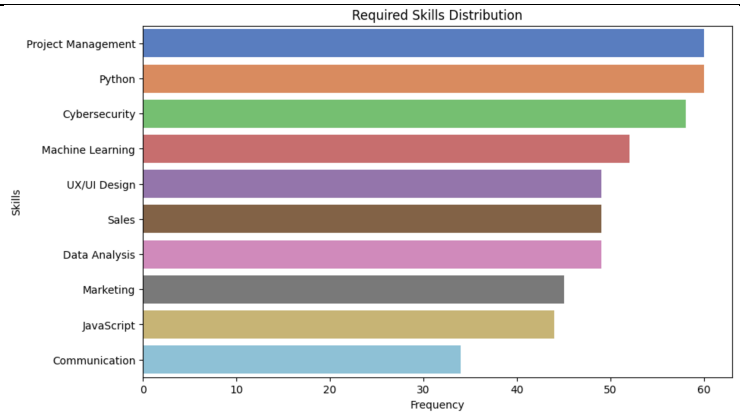
The variance of 420,425,968.01 indicates a substantial spread in the salary data, highlighting significant variability between the lowest and highest salaries. This further emphasizes the presence of outliers and the diverse nature of salaries in the dataset, which may require normalization to improve data consistency for analysis.

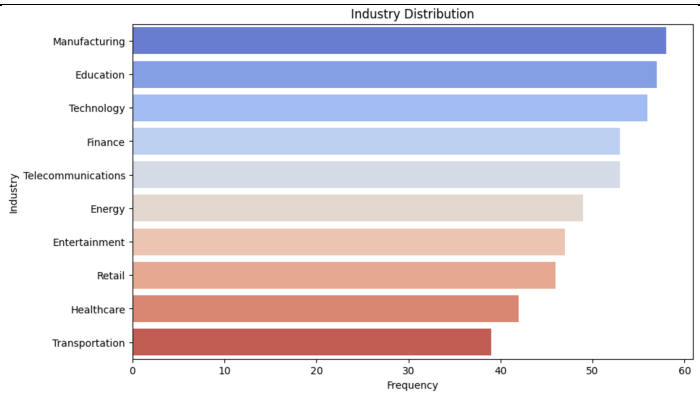
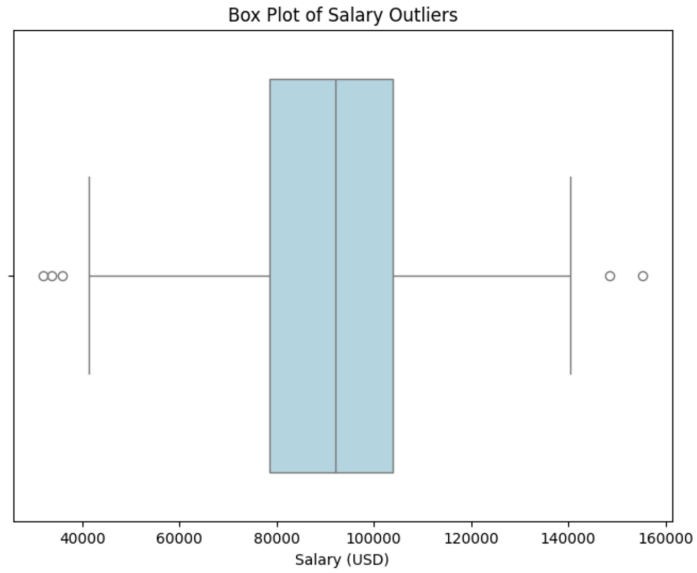
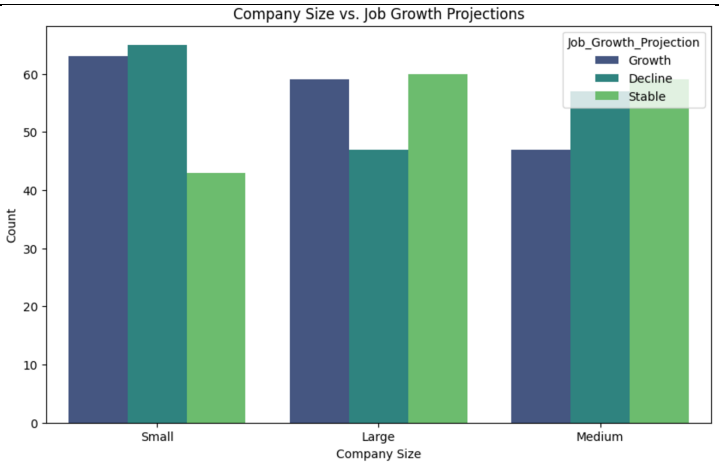
Understanding the data through graph representations:

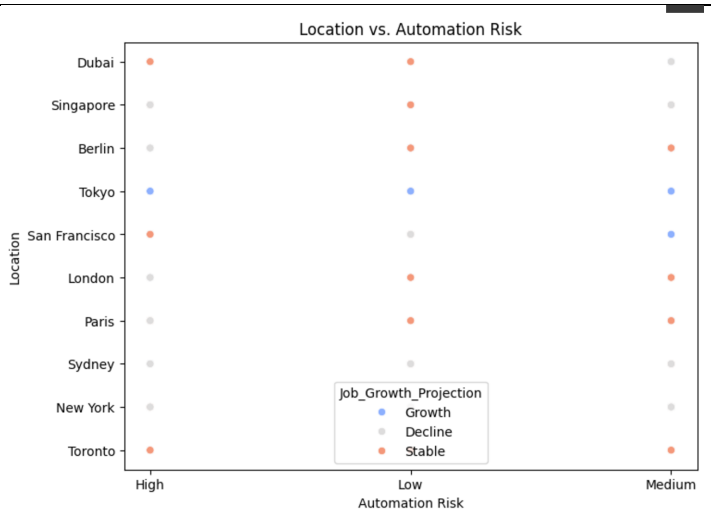
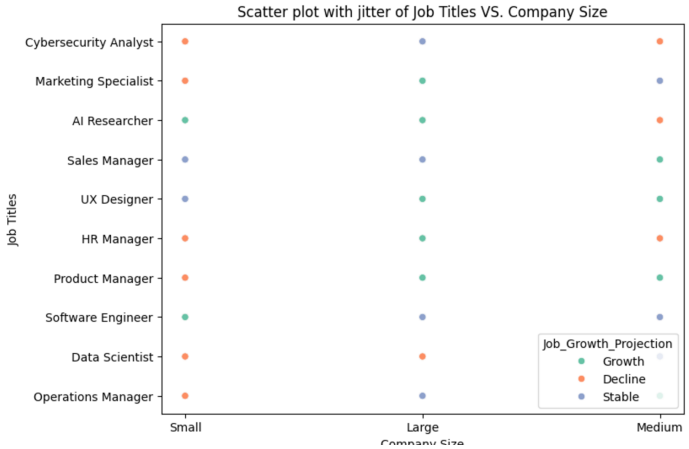
To analyze the factors influencing job growth projections across various industries, the dataset was explored through a series of graph representations. These visualizations reveal relationships between attributes such as salary, automation risk, AI adoption levels, and required skills, and their impact on job

growth trends (growth, decline, or stable). By focusing on the class label, Job Growth Projection, the analysis highlights the industries and job roles most likely to experience growth or decline. The insights derived from these graphs enable us to better understand the evolving job market and the role of automation and AI adoption in shaping the future of employment opportunities.

Name of Graph	Picture of Graph	Description
Pie Chart		**The pie charts compare the job growth projections for remote-friendly and non-remote-friendly roles. Both categories show similar proportions of growth, decline, and stability, suggesting no significant difference in trends based on remote compatibility.**
Pie Chart		**These pie charts illustrate the job growth projection distribution across AI adoption levels (Low, Medium, High). Each chart shows the proportion of roles projected to experience growth, decline, or remain stable, highlighting that higher AI adoption correlates with a slightly increased likelihood of growth.**

Histogram		**The histogram visualizes the distribution of salaries in USD across job roles. Most salaries fall between $70,000 and $110,000, indicating that the majority of jobs in the dataset offer mid-range salaries, with fewer roles at the extreme high or low ends.**
Scatter Plot		**The scatter plot illustrates the relationship between automation risk levels (High, Low, Medium) and salaries in USD. Data points are categorized by job growth projections (Growth, Decline, Stable). It reveals no clear trend between automation risk and salary, with salaries distributed across all levels of automation risk.**
Bar Chart		**This bar chart displays the distribution of required skills across job roles. The x-axis represents the frequency of each skill, and the y-axis lists the skills. It highlights the most in-demand skills, such as Project Management and Python, which appear most frequently in job descriptions.**

Bar Chart		**This bar chart illustrates the distribution of job roles across various industries. The x-axis represents the frequency of jobs in each industry, while the y-axis lists the industries. The chart highlights which sectors, such as Manufacturing and Education, have the highest representation in the dataset.**
Box Plot		**This box plot visualizes the distribution of salaries (USD) in the dataset, highlighting the interquartile range, median, and outliers. The central box represents the middle 50% of salaries, while the whiskers extend to the non-outlier minimum and maximum values. Outliers, represented as individual points, indicate salaries significantly higher or lower than the majority.**
bar chart		**This bar chart shows the distribution of job growth projections (Growth, Decline, Stable) across different company sizes (Small, Medium, Large). It highlights how job growth projections vary based on the scale of the organization, offering insights into employment trends related to company size.**

scatter plot		**This scatter plot visualizes the relationship between job locations and their associated automation risk levels (High, Low, Medium), color-coded by job growth projection (Growth, Decline, Stable). It highlights how automation risk varies across different cities and its potential impact on job growth trends.**
Scatter plot		**This scatter plot shows the distribution of job titles across company sizes (Small, Medium, Large) with points color-coded by job growth projections (Growth, Decline, Stable). It provides insights into how different roles are represented within various company sizes and their future growth potential.**

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


#Scatter plot of Job Titles VS. Company Size
plt.figure(figsize=(8,6))
sns.scatterplot(x='Company_Size', y='Job_Title', data=df, hue='Job_Growth_Projection',palette='Set2')
plt.title('Scatter plot with jitter of Job Titles VS. Company Size')
plt.xlabel('Company Size')
plt.ylabel ('Job Titles')
plt.show()
plt.subplots_adjust(hspace=0.5)
```

```
# Histogram for 'Salary_USD'
plt.figure(figsize=(8,6))
df['Salary_USD'].hist(bins=30)
plt.title('Histogram of Salary')
plt.xlabel('Salary (USD)')
plt.ylabel('Frequency')
plt.show()
```

```
#pie chart Remote Friendly
yes_counts = df[df['Remote_Friendly'] == 'Yes']['Job_Growth_Projection'].value_counts()
no_counts = df[df['Remote_Friendly'] == 'No']['Job_Growth_Projection'].value_counts()

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.pie(yes_counts, labels=yes_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Job Growth Projection - Remote_Friendly: Yes')

plt.subplot(1, 2, 2)
plt.pie(no_counts, labels=no_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Job Growth Projection - Remote Friendly: No')
plt.subplots_adjust(hspace=0.5)

plt.show()
```

```
#bar chart Required_Skills
plt.figure(figsize=(8,6))
sns.countplot(y='Required_Skills',data=df)
plt.title('Distribution of Required_Skills')
plt.show()

plt.subplots_adjust(hspace=0.5)
```

```
#Scatter plot with jitter of Location VS. Automation_Risk
plt.figure(figsize=(8,6))
sns.scatterplot(x='Automation_Risk', y='Location', data=df, hue='Job_Growth_Projection',palette='Set2')
plt.title('Scatter plot with jitter of Automation_Risk VS. Location')
plt.ylabel('Location')
plt.xlabel ('Automation_Risk')
plt.show()

plt.subplots_adjust(hspace=0.5)
```

```
#bar chart Industry
plt.figure(figsize=(8,6))
sns.countplot(y='Industry',data=df)
plt.title('Distribution of Industry')
plt.show()

plt.subplots_adjust(hspace=0.5)
```

```
#pie chart Ai_adoption_level
levels = ['Low', 'Medium', 'High']
job_growth_counts = {level: df[df['AI_Adoption_Level'] == level]['Job_Growth_Projection'].value_counts() for level in level

plt.figure(figsize=(15, 5))

for i, level in enumerate(levels):
    if level in job_growth_counts:
        plt.subplot(1, 3, i + 1)
        plt.pie(job_growth_counts[level], labels=job_growth_counts[level].index, autopct='%1.1f%%', startangle=140)
        plt.title(f'Job Growth Projection - AI Adoption Level: {level}')
    else:
        print(f"No data for {level}")

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

4- Data preprocessing:

Checking for missing values:

```
Missing values in each column:
Job_Title                0
Industry                 0
Company_Size             0
Location                 0
AI_Adoption_Level        0
Automation_Risk          0
Required_Skills          0
Salary_USD               0
Remote_Friendly          0
Job_Growth_Projection    0
```

Description:

The dataset was examined for missing or null values across all columns to ensure its completeness. Upon analysis, no missing values were identified, indicating that the dataset is of high quality and does not require additional imputation or handling of missing data. This

ensures that no information is lost or distorted during preprocessing, allowing for a straightforward analysis.

Detecting outliers:

```
#In our project analyzing how AI and automation impact employment, we used outlier detection to improve data accuracy

outlier_threshold = 1.5

def count_outliers(column_data):
    q1 = np.percentile(column_data, 25)
    q3 = np.percentile(column_data, 75)
    iqr = q3 - q1
    upper_bound = q3 + outlier_threshold * iqr
    lower_bound = q1 - outlier_threshold * iqr
    outliers = (column_data > upper_bound) | (column_data < lower_bound)
    return sum(outliers)

numeric_columns = data.select_dtypes(include=[np.number]).columns

outlier_counts = {}
total_rows_with_outliers = 0

for column in numeric_columns:
    outliers = count_outliers(data[column])
    outlier_counts[column] = outliers
    total_rows_with_outliers += outliers

total_rows = len(data)

print("Outlier Counts:")
for column, count in outlier_counts.items():
    print(f"{column}: {count} rows with outliers")

print(f"Total Rows with Outliers: {total_rows_with_outliers}")
```

```
Outlier Counts:
Salary_USD: 5 rows with outliers
Total Rows with Outliers: 5
```

Description:

During data preprocessing, the Salary_USD attribute was analyzed for outliers. A total of 5 rows were identified as outliers in this column. These outliers represent extreme salary values that deviate significantly from the rest of the data. Outlier detection is essential to ensure that these extreme values do not skew the analysis or modeling results. Depending on the context, these outliers may either be retained for further analysis or handled to enhance the dataset's reliability.

Removing Outliners:

```
# Calculate the IQR (Interquartile Range)
Q1 = data['Salary_USD'].quantile(0.25) # Calculate the first quartile (25th percentile)
Q3 = data['Salary_USD'].quantile(0.75) # Calculate the third quartile (75th percentile)
IQR = Q3 - Q1

# Define the bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the data to remove outliers
cleaned_data = data[(data['Salary_USD'] >= lower_bound) & (data['Salary_USD'] <= upper_bound)]


# Display the removed rows to understand what was filtered out
removed_rows = data[(data['Salary_USD'] < lower_bound) | (data['Salary_USD'] > upper_bound)]
print("\nRemoved Rows (Outliers):")
print(removed_rows)
```

Raw data:

```
  AI_Adoption_Level Automation_Risk     Required_Skills     Salary_USD  \
0            Medium            High        UX/UI Design  111392.165243
1            Medium            High           Marketing   93792.562466
2            Medium            High        UX/UI Design  107170.263069
3               Low            High  Project Management   93027.953758
4               Low             Low          JavaScript   87752.922171

  Remote_Friendly Job_Growth_Projection
0             Yes                Growth
1              No               Decline
2             Yes                Growth
3              No                Growth
4             Yes               Decline
```

After processing:

```
Cleaned Dataset (Without Outliers):
             Job_Title       Industry Company_Size   Location  \
0  Cybersecurity Analyst  Entertainment        Small      Dubai
1   Marketing Specialist     Technology        Large  Singapore
2         AI Researcher     Technology        Large  Singapore
3         Sales Manager         Retail        Small     Berlin
4  Cybersecurity Analyst  Entertainment        Small      Tokyo

  AI_Adoption_Level Automation_Risk     Required_Skills     Salary_USD  \
0            Medium            High        UX/UI Design  111392.165243
1            Medium            High           Marketing   93792.562466
2            Medium            High        UX/UI Design  107170.263069
3               Low            High  Project Management   93027.953758
4               Low             Low          JavaScript   87752.922171

  Remote_Friendly Job_Growth_Projection
0             Yes                Growth
1              No               Decline
2             Yes                Growth
3              No                Growth
4             Yes               Decline

Removed Rows (Outliers):
               Job_Title            Industry Company_Size       Location  \
182         Data Scientist      Transportation        Small       New York
289         Data Scientist          Healthcare       Medium          Paris
384  Cybersecurity Analyst  Telecommunications        Large         Berlin
420   Marketing Specialist             Finance       Medium  San Francisco
425          UX Designer       Entertainment        Small      Singapore

    AI_Adoption_Level Automation_Risk Required_Skills     Salary_USD  \
182               Low             Low          Python   31969.526346
289              High             Low          Python  148467.112346
384               Low          Medium    UX/UI Design   33601.381360
420              High            High           Sales  155209.821614
425            Medium          Medium   Data Analysis   35963.297317
```

Description:

These rows were identified as outliers due to their salary values falling significantly below or above the acceptable range defined by the IQR method. By filtering out these extreme values, the resulting dataset becomes more representative of the typical salary distribution, leading to more accurate and reliable analysis.

Data Transformation:

1. Encoding

```
# Redefining the necessary variables to apply encoding directly to the original dataframe
columns_to_encode = ['Job_Title', 'Industry', 'Company_Size', 'Location',
                     'AI_Adoption_Level', 'Automation_Risk', 'Required_Skills',
                     'Remote_Friendly', 'Job_Growth_Projection']

# Re-initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply the encoding directly to the original dataframe without creating a copy
for column in columns_to_encode:
    df[column] = label_encoder.fit_transform(df[column])

# Display the first few rows of the modified original dataframe to verify the changes
df.head()
```

Raw data:

```
Data Before Encoding:
        Job_Title    Industry Company_Size  Location AI_Adoption_Level  \
0  Data Scientist  Technology        Large  New York              High
1   AI Researcher     Finance       Medium    London            Medium
2   Sales Manager      Retail        Small     Dubai               Low

  Automation_Risk Required_Skills Remote_Friendly Job_Growth_Projection
0            High          Python             Yes                Growth
1             Low   Data Analysis              No                Stable
2          Medium           Sales             Yes               Decline
```

After processing:

	Job_Title	Industry	Company_Size	Location	AI_Adoption_Level	Automation_Risk	Required_Skills	Salary_USD	Remote_Friendly
0	1	2	2	1	2	0	9	111392.165243	1
1	4	7	0	6	2	0	5	93792.562466	0
2	0	7	0	6	2	0	9	107170.263069	1
3	7	6	2	0	1	0	6	93027.953758	0
4	1	2	2	8	1	1	3	87752.922171	1

Description:

To prepare the data for machine learning, categorical attributes such as Job_Title, Industry, and Required_Skills were encoded into numerical values using Label Encoding. This approach assigns a unique integer to each category, ensuring compatibility with machine learning models while preserving the categorical relationships within the data. The transformation helps to standardize the dataset for further analysis.

2.Normalization:

```
columns_to_normalize = ['Job_Title','Industry','Company_Size' ,'Location','AI_Adoption_Level', 'Automation_Risk','Required_
# Removed columns containing string values: 'Industry', 'Company_Size', 'Location', 'Required_Skills'


# Decimal scaling normalization
for column in columns_to_normalize:
    max_abs_value = df[column].abs().max()
    df[column] = df[column] / (10 ** len(str(int(max_abs_value))))

print("DataFrame after Decimal Scaling Normalization:")
print(df)
```

Raw data:

```
DataFrame Before Normalization:
   Job_Title  Industry  Company_Size  Location  AI_Adoption_Level  \
0        0.1       0.2           0.0       0.2                0.0
1        0.0       0.0           0.1       0.1                0.2
2        0.2       0.1           0.2       0.0                0.1

   Automation_Risk  Required_Skills  Remote_Friendly  Job_Growth_Projection
0              0.0              0.1                1                     1
1              0.1              0.0                0                     2
2              0.2              0.2                1                     0
```

After processing:

```
DataFrame after Decimal Scaling Normalization:
     Job_Title  Industry  Company_Size  Location  AI_Adoption_Level  \
0          0.1       0.2           0.2       0.1                0.2
1          0.4       0.7           0.0       0.6                0.2
2          0.0       0.7           0.0       0.6                0.2
3          0.7       0.6           0.2       0.0                0.1
4          0.1       0.2           0.2       0.8                0.1
..         ...       ...           ...       ...                ...
495        0.2       0.8           0.1       0.0                0.1
496        0.1       0.8           0.2       0.2                0.1
497        0.1       0.1           0.0       0.1                0.0
498        0.5       0.4           0.0       0.4                0.0
499        0.3       0.2           0.1       0.0                0.2

     Automation_Risk  Required_Skills  Salary_USD  Remote_Friendly  \
0                0.0              0.9    0.111392                1
1                0.0              0.5    0.093793                0
2                0.0              0.9    0.107170                1
3                0.0              0.6    0.093028                0
4                0.1              0.3    0.087753                1
..               ...              ...         ...              ...
495              0.2              0.4    0.105821                1
496              0.0              0.9    0.119795                0
497              0.1              0.9    0.079645                1
498              0.1              0.7    0.077642                1
499              0.0              0.6    0.068764                1

     Job_Growth_Projection
0                      0.1
1                      0.0
2                      0.1
3                      0.1
4                      0.0
..                     ...
495                    0.2
496                    0.0
497                    0.2
498                    0.2
499                    0.0
```

Description:

Before normalization, the dataset had attributes with varying scales, such as large Salary_USD values compared to encoded categorical data. After applying decimal scaling normalization, all attributes were scaled to comparable ranges, ensuring fair contribution to analysis and improving model performance.

3.Discrization

```
bins = [0, 50000, 100000, 150000, 200000]
labels = ['Low', 'Medium', 'High', 'Very High']
data['Salary_Category'] = pd.cut(data['Salary_USD'], bins=bins, labels=labels)  # Categorizing salaries
print("After Discretization (Salary Categories):")
print(data[['Salary_USD', 'Salary_Category']].head(), "\n")
```

Raw data:

Salary_USD
111392.165243
93792.562466
107170.263069
93027.953758
87752.922171

After processing:

```
After Discretization (Salary Categories):
      Salary_USD Salary_Category
0  111392.165243            High
1   93792.562466          Medium
2  107170.263069            High
3   93027.953758          Medium
4   87752.922171          Medium
```

Description:

we categorize the salary data into discrete ranges to better understand the distribution of salaries. We define bins that represent salary ranges: [0, 50000], [50000, 100000], [100000, 150000], and [150000, 200000]. These bins correspond to the labels 'Low', 'Medium', 'High', and 'Very High'.

5. Data Mining Technique

We utilized both supervised and unsupervised learning techniques to analyze our dataset through classification and clustering methods.

For the classification task, we implemented a decision tree. This recursive algorithm builds a tree structure where each branch represents a decision rule, and each leaf node corresponds to a final classification. Our model predicts the job growth projection (e.g., growth, stable, decline) using multiple attributes, including

Salary_USD, Automation_Risk, and AI_Adoption_Level. Classification, being a supervised learning approach, required us to split the dataset into training and testing subsets. We experimented with training sizes of 70%, 60%, and 80% to evaluate the impact of data partitioning on model performance. Attribute selection was optimized using measures like Information Gain (entropy) and Gini Index. To validate the model's effectiveness, we used a confusion matrix to assess performance metrics such as accuracy, sensitivity, specificity, and precision.

For the clustering task, we employed the K-means algorithm, which is an unsupervised learning technique. Unlike classification, clustering does not rely on labeled data. We excluded the Job_Growth_Projection attribute and used the remaining features, including Salary_USD, Automation_Risk, and AI_Adoption_Level. The K-means algorithm grouped the data into K clusters by minimizing intra-cluster variance. Each object was assigned to the nearest cluster, and centroids were iteratively recalculated until stabilization. To determine the optimal number of clusters, we used the Within-Cluster Sum of Squares (WSS) method and evaluated cluster quality with silhouette scores. These metrics ensured compactness and separation of clusters.

Through these approaches, we combined the strengths of supervised and unsupervised learning to derive meaningful insights from the data. Classification enabled precise predictions, while clustering revealed hidden patterns and groupings within the dataset.

6. Evaluation and Comparison

```
# Define features (X) and target (y)
X = preprocessed_data[['Job_Title', 'Industry', 'Company_Size', 'Location',
                       'AI_Adoption_Level', 'Automation_Risk', 'Required_Skills',
                       'Salary_USD', 'Remote_Friendly']]
y = preprocessed_data['Job_Growth_Projection']


#splitting to ratios
splits = [(0.9, 0.1), (0.8, 0.2), (0.7, 0.3)]

#Loop through the splits and evaluate
for train_size, test_size in splits:
    print(f"\n\n==== Data split: {int(train_size*100)}%-{int(test_size*100)}% ====")

    #splitting the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=train_size, test_size=test_size, random_state=42)

    for criterion in ['gini', 'entropy']:  #default is gini
        print(f"\n--- Using {criterion.upper()} as the criterion ---")

        #training the Decision Tree
        clf = DecisionTreeClassifier(criterion=criterion, random_state=42)
        clf.fit(X_train, y_train)

        #predictions and evaluation
        y_pred = clf.predict(X_test)
        print("Confusion Matrix:")
        print(confusion_matrix(y_test, y_pred))
        print("\nClassification Report:")
        print(classification_report(y_test, y_pred, target_names=['Decline', 'Stable', 'Growth']))

        #Visualizing the Decision Tree
        plt.figure(figsize=(12, 8))
        plot_tree(clf, filled=True, feature_names=X.columns, class_names=['Decline', 'Stable', 'Growth'], max_depth=5)
        plt.title(f"Decision Tree ({criterion.capitalize()}) - Split {int(train_size*100)}%-{int(test_size*100)}%")
        plt.show()
```

- **Classification [90% training, 10% testing] Information Gain:**

Figure (1) (decision tree):

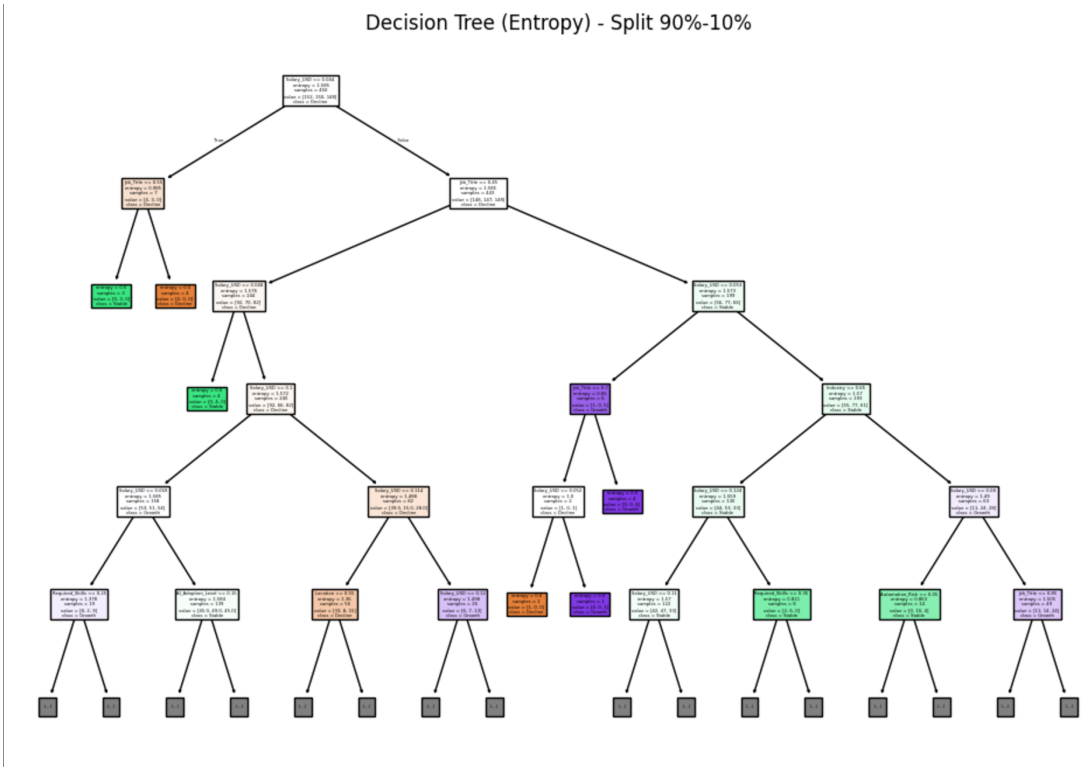
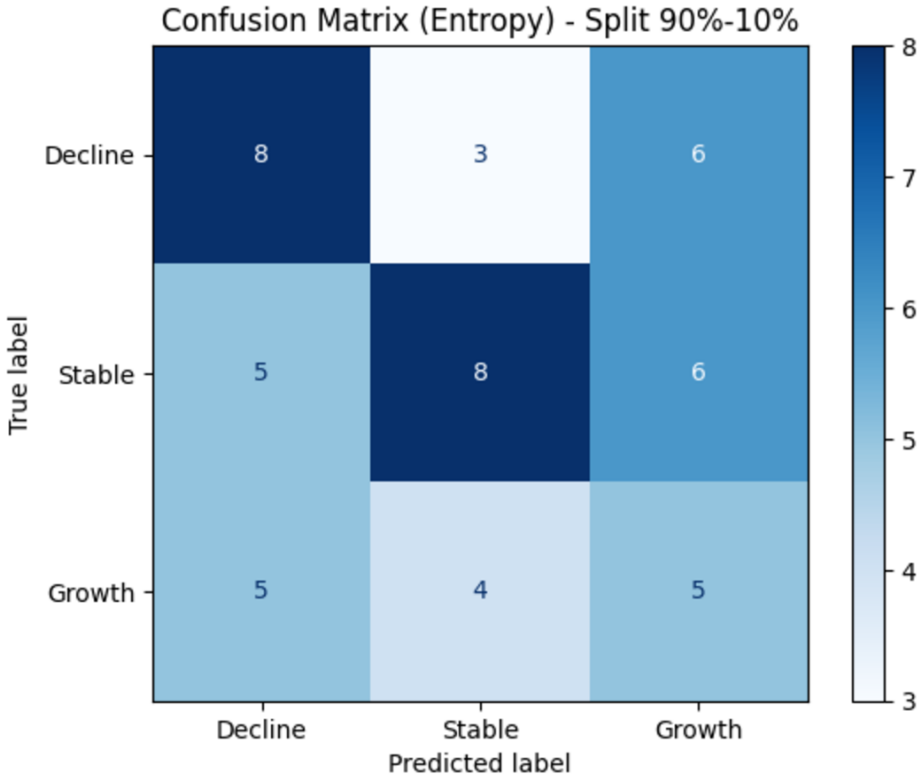


Figure (2) (confusion matrix):



● Classification [80% training, 20% testing] Information Gain:

Figure (1) (decision tree):

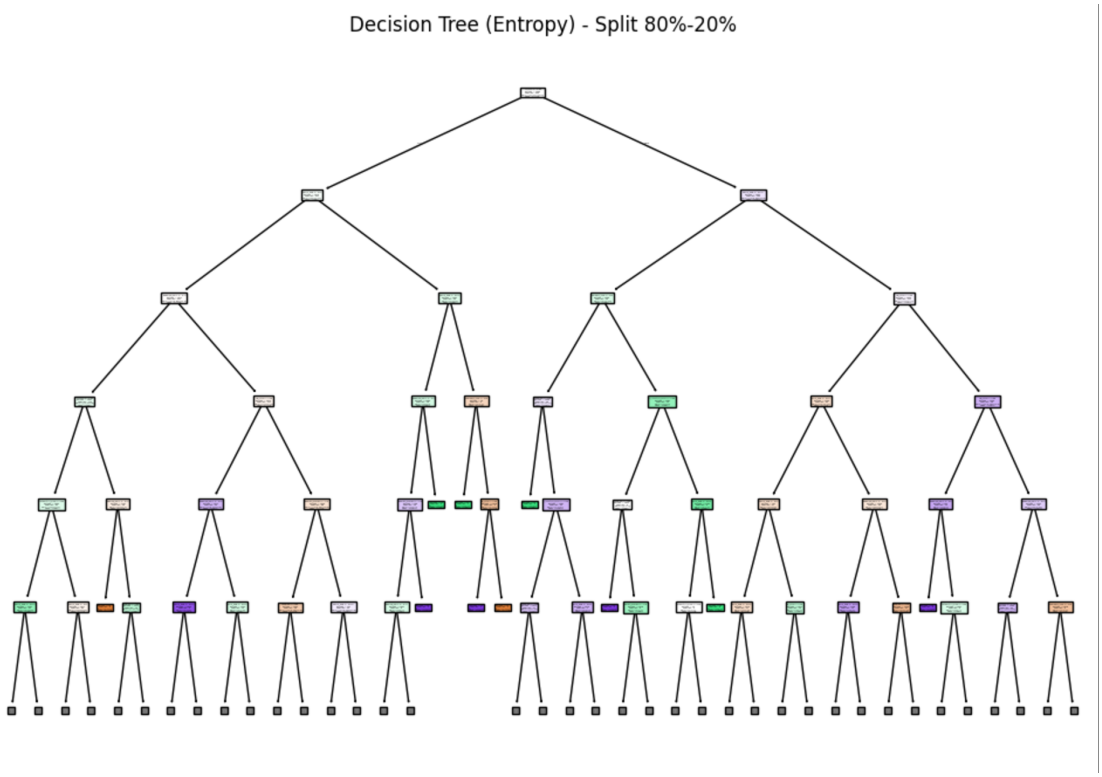
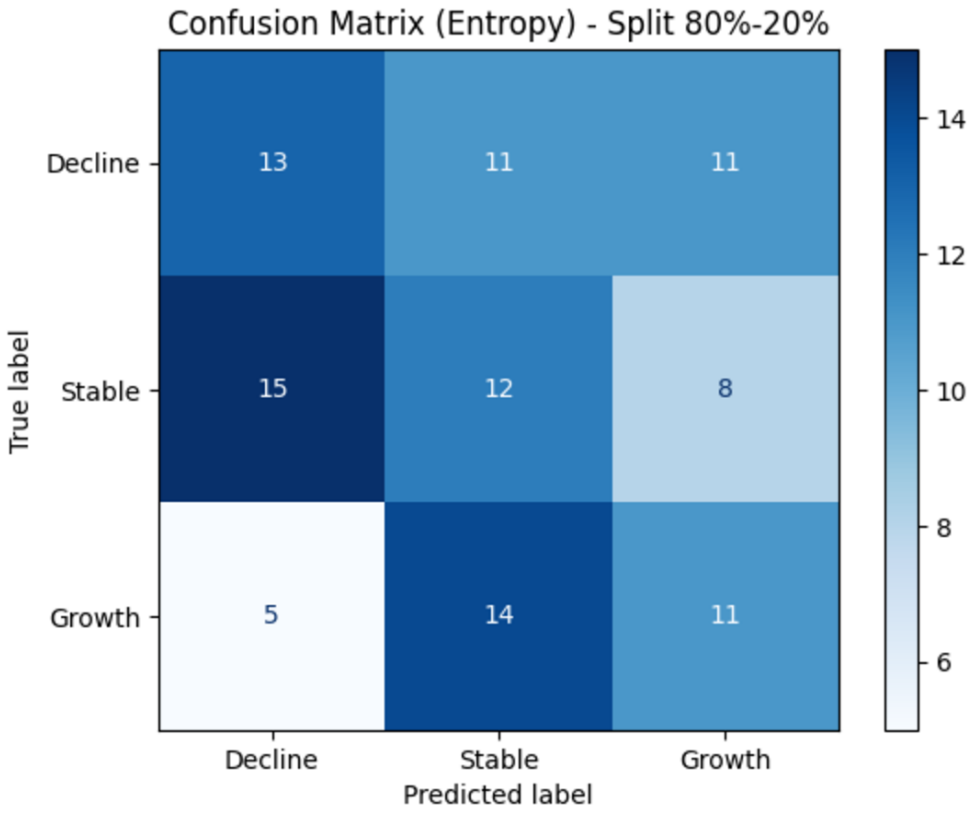


Figure (2) (confusion matrix):



● Classification [70% training, 30% testing] Information Gain:

Figure (1) (decision tree):

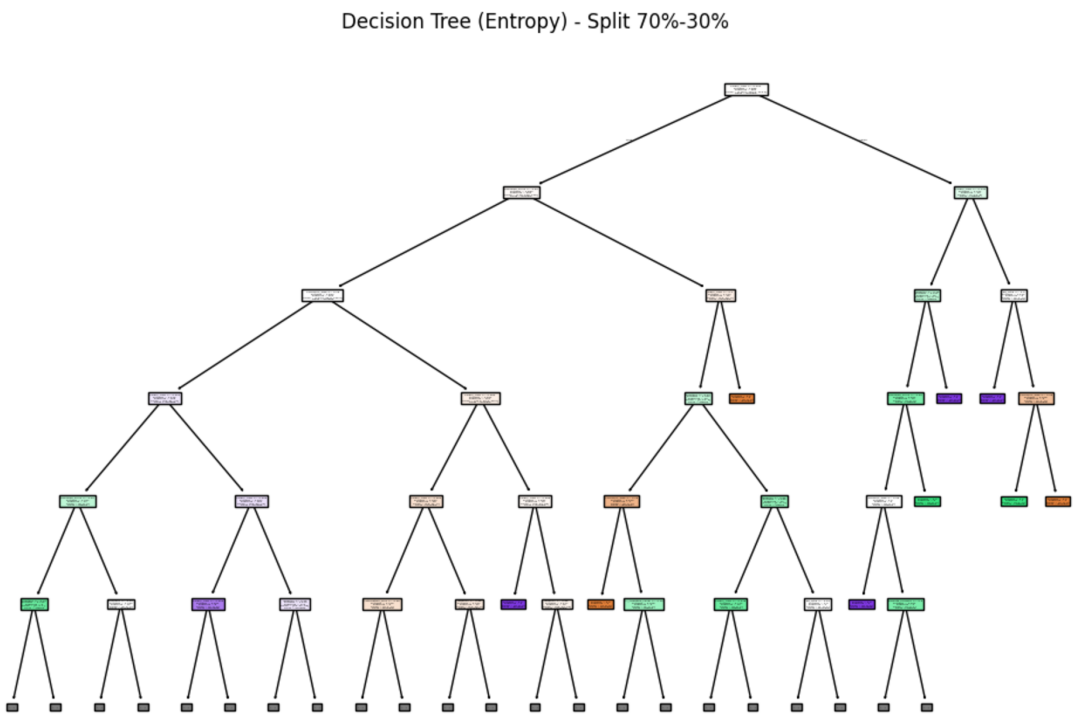
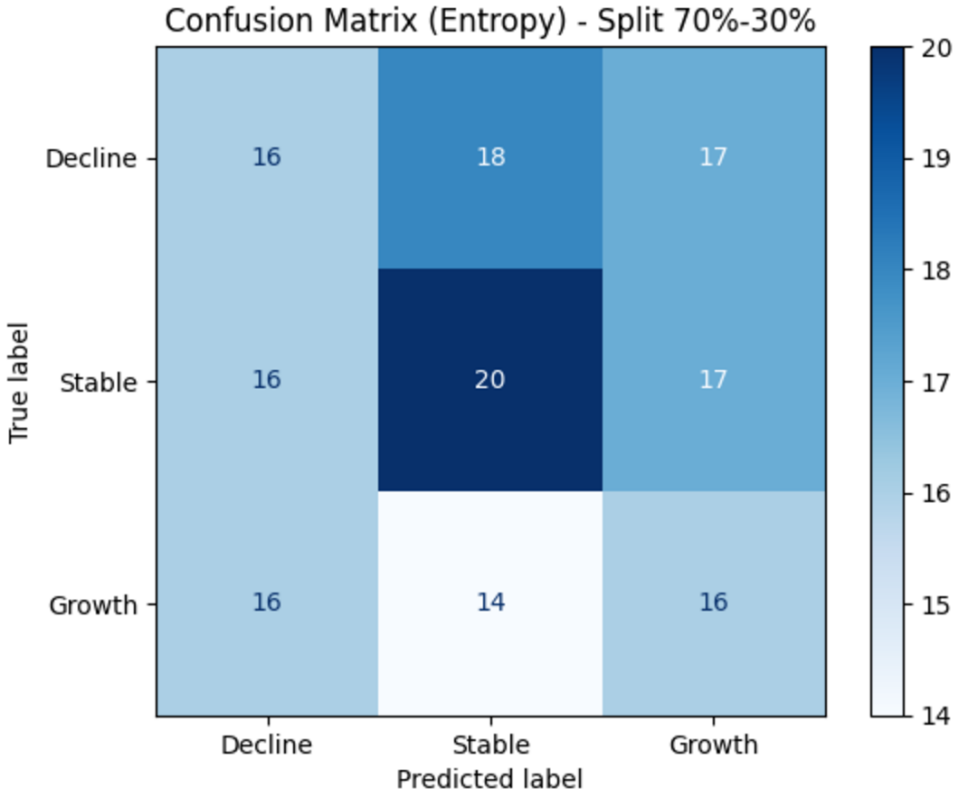


Figure (2) (confusion matrix):



<table>
<tr><th>Mining task</th><th>Comparison Criteria</th></tr>
<tr><td>Classification for Information Gain</td><td>
<table>
<tr><th></th><th>90 %training set 10% testing set:</th><th>80 % training set 20% testing set:</th><th>70 % training set 30% testing set:</th></tr>
<tr><td>Accuracy</td><td>42%</td><td>36%</td><td>35%</td></tr>
<tr><td>precision</td><td>44%</td><td>36%</td><td>35%</td></tr>
<tr><td>sensitivity</td><td>42%</td><td>36%</td><td>35%</td></tr>
<tr><td>specificity</td><td>51%</td><td>46%</td><td>45%</td></tr>
<tr><td>Error rate</td><td>58%</td><td>64%</td><td>65%</td></tr>
</table>
</td></tr>
</table>

● **Classification [90% training, 10% testing] Gini Index :**

Figure (1) (decision tree):

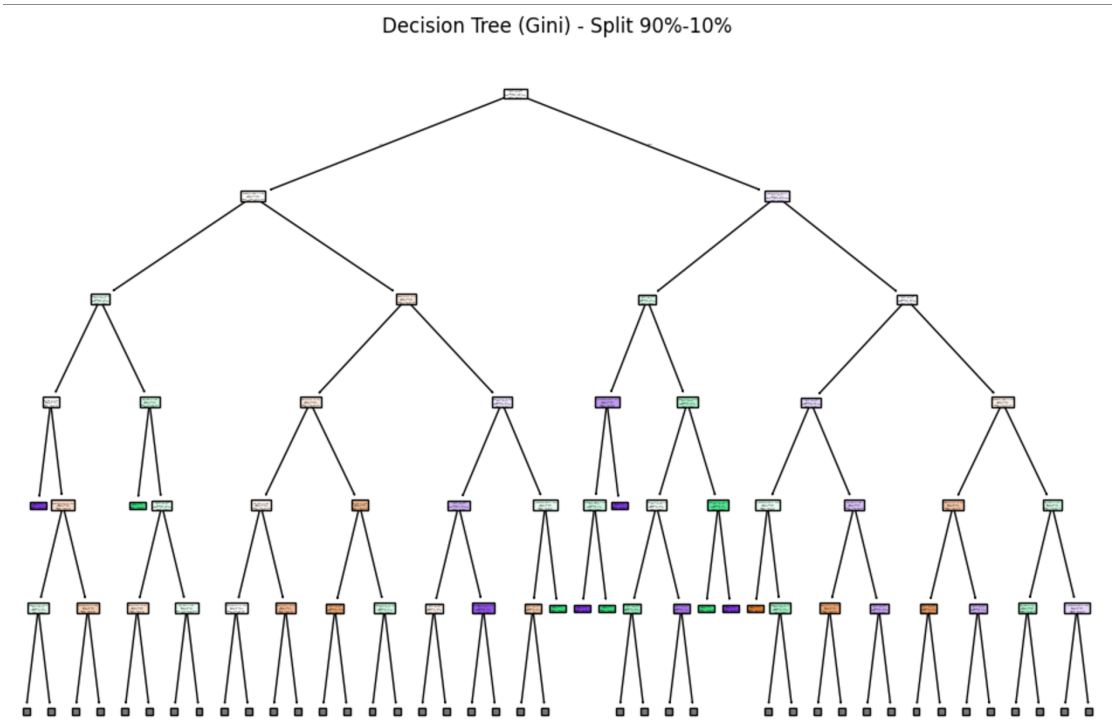
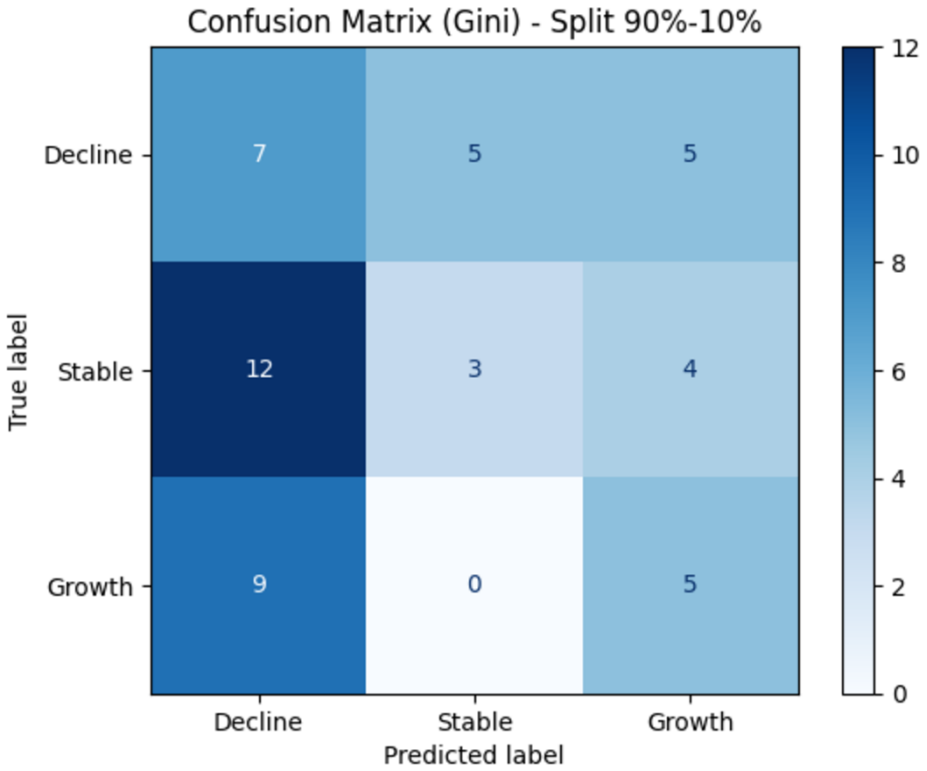


Figure (2) (confusion matrix):



● **Classification [80% training, 20% testing] Gini Index :**

Figure (1) (decision tree):

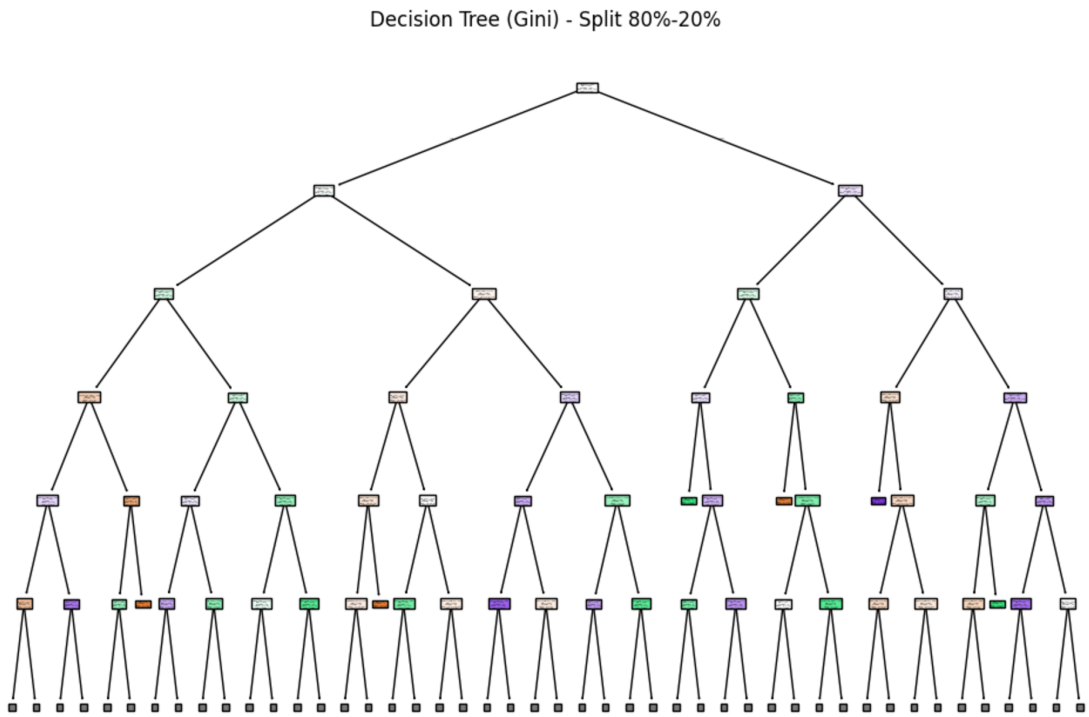
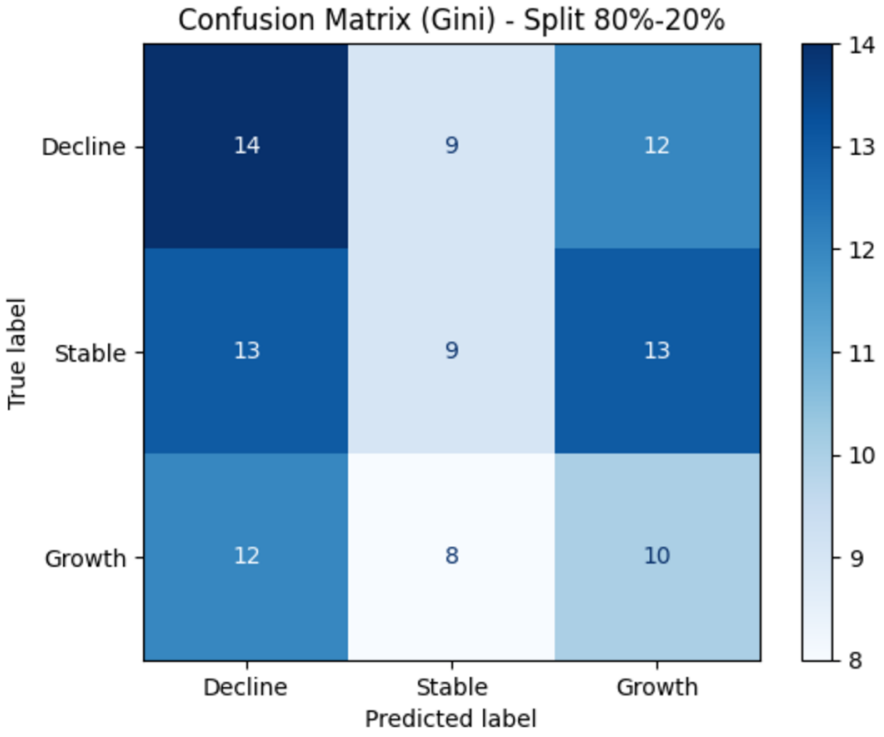


Figure (2) (confusion matrix):



● **Classification [70% training, 30% testing] Gini Index :**

Figure (1) (decision tree):

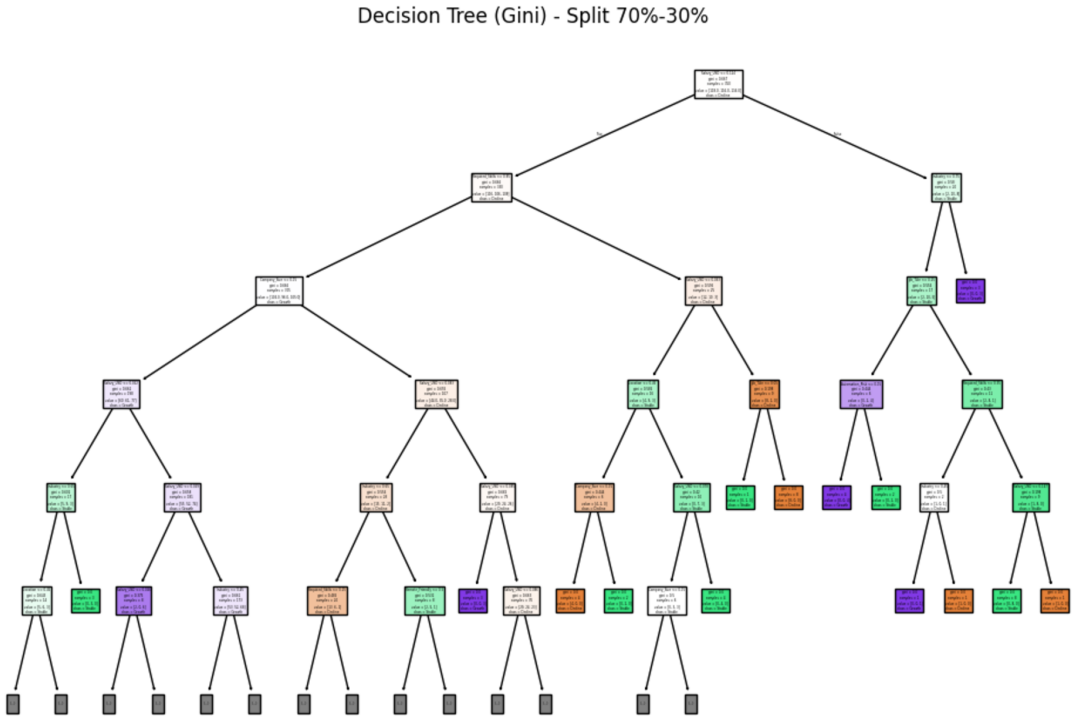
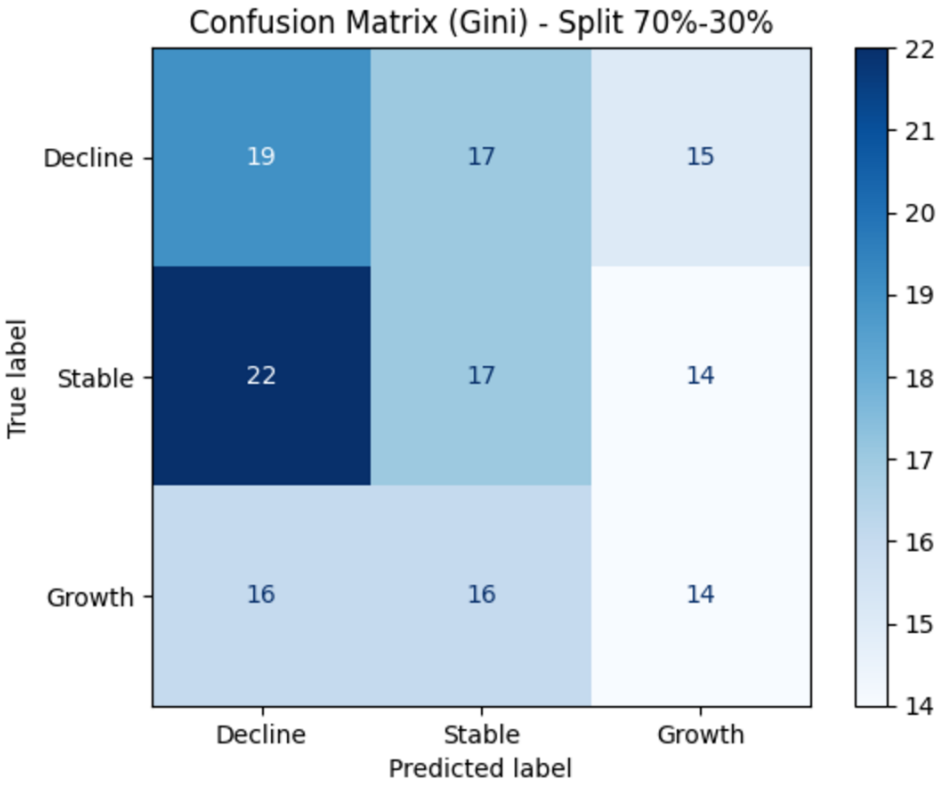


Figure (2) (confusion matrix):



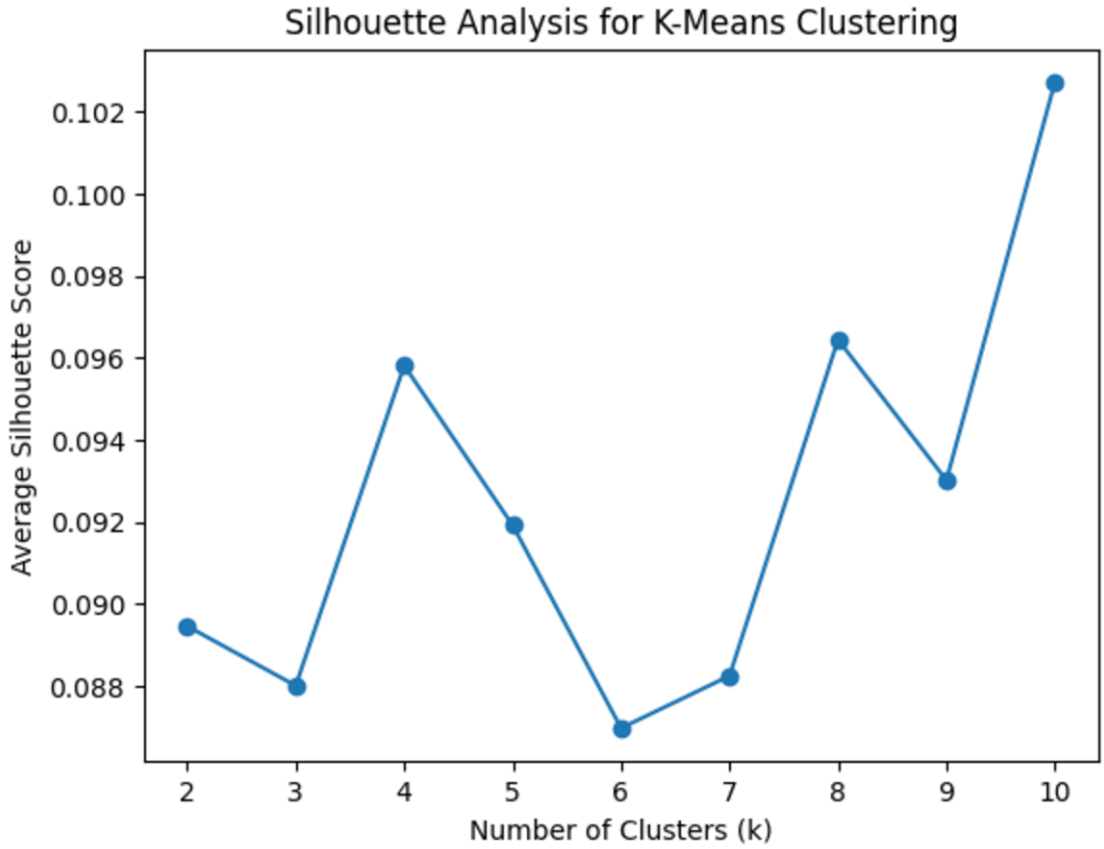
<table>
<tr><th>Mining task</th><th>Comparison Criteria</th></tr>
<tr><td>Classification for Gini Index</td><td>
<table>
<tr><th></th><th>90 %training set 10% testing set:</th><th>80 % training set 20% testing set:</th><th>70 % training set 30% testing set:</th></tr>
<tr><td>Accuracy</td><td>30%</td><td>33%</td><td>33%</td></tr>
<tr><td>precision</td><td>33%</td><td>33%</td><td>33%</td></tr>
<tr><td>sensitivity</td><td>30%</td><td>33%</td><td>33%</td></tr>
<tr><td>specificity</td><td>49%</td><td>48%</td><td>46%</td></tr>
<tr><td>Error rate</td><td>70%</td><td>67%</td><td>67%</td></tr>
</table>
</td></tr>
</table>

The 90%-10% split with Information Gain emerges as the better partitioning strategy due to its better accuracy, precision, and error rate compared to the other configurations.

Clustering

the choice for the three different sizes of K-means clustering:

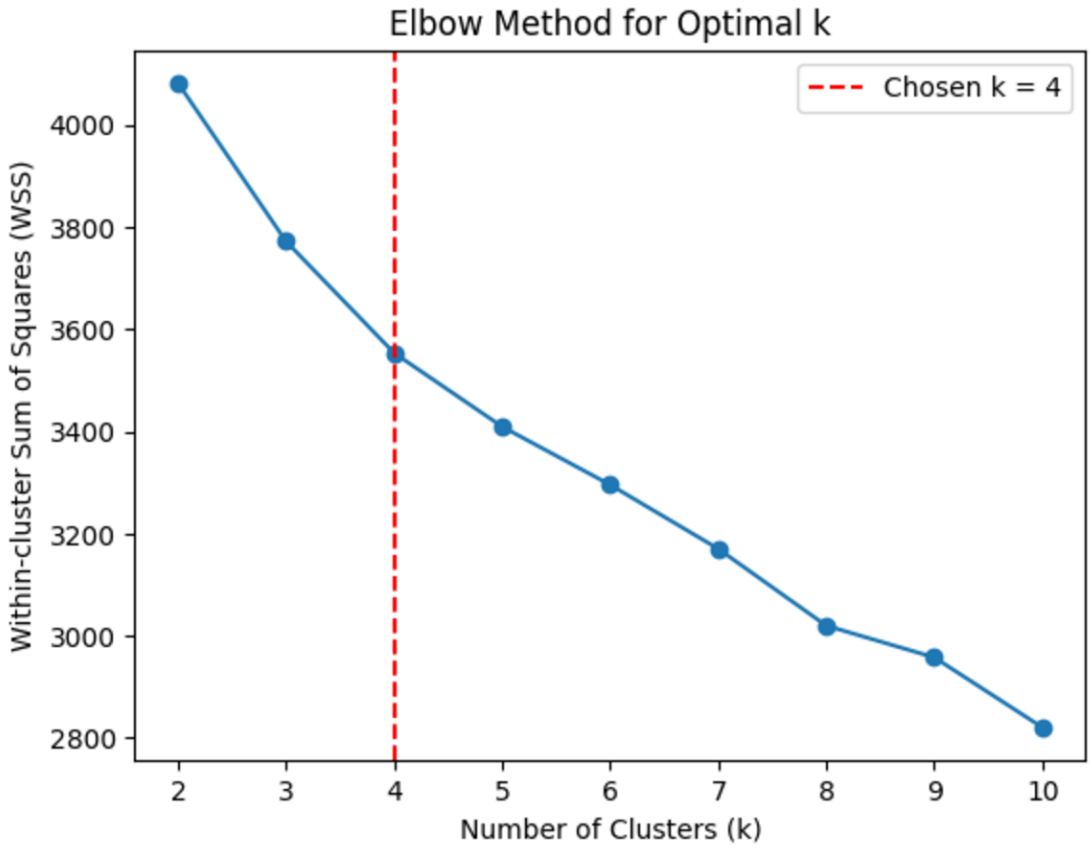
-Silhouette method




```
# Define the range for k values
k_values = range(2, 11)
silhouette_avg_values = []

# Perform K-means clustering and calculate silhouette scores
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    labels = kmeans.fit_predict(df_scaled)
    silhouette_avg = silhouette_score(df_scaled, labels)
    silhouette_avg_values.append(silhouette_avg)

# Plot silhouette scores for each k
plt.plot(k_values, silhouette_avg_values, marker='o')
plt.title('Silhouette Analysis for K-Means Clustering')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Average Silhouette Score')
plt.show()

# Find the best and second-best k values
best_k = k_values[np.argmax(silhouette_avg_values)]
best_score = max(silhouette_avg_values)
silhouette_avg_values_sorted = sorted(silhouette_avg_values, reverse=True)
second_best_score = silhouette_avg_values_sorted[1]
second_best_k = k_values[silhouette_avg_values.index(second_best_score)]

print(f"The highest average Silhouette score is {best_score} with k={best_k}.")
print(f"The second highest average Silhouette score is {second_best_score} with k={second_best_k}.")
```

In the silhouette plot, we observed that the highest average silhouette score was obtained with k=10 (score of 0.1027), followed by k=8 (score of 0.0964). These scores indicate that k=10 provides the most distinct clusters based on the silhouette analysis, suggesting that 10 clusters may be optimal for our data in terms of cluster cohesion and separation.

Elbow method

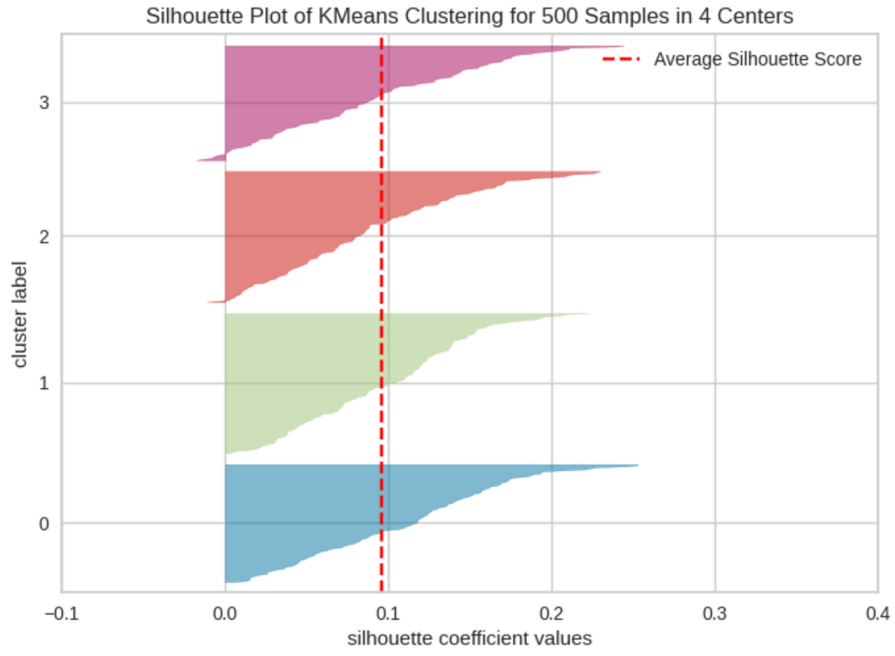


This suggests that k=4 is a suitable choice, as it represents a balance between cluster compactness and simplicity. Beyond k=4, the decrease in WSS becomes more gradual, indicating that additional clusters provide diminishing returns in terms of clustering quality. Therefore, we selected k=4 as the optimal number of clusters based on the Elbow Method.

```
!pip install kneed

import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from kneed import KneeLocator

# Assuming df_scaled is already scaled
k_values = range(2, 11)
wss_values = []

# Calculate WSS for each k
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    kmeans.fit(df_scaled)
    wss_values.append(kmeans.inertia_)

# Plot the elbow method
plt.plot(k_values, wss_values, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Within-cluster Sum of Squares (WSS)')
plt.title('Elbow Method for Optimal k')

# Use KneeLocator to find the elbow point
knee = KneeLocator(k_values, wss_values, curve='convex', direction='decreasing')
turning_point = knee.elbow

plt.axvline(x=turning_point, linestyle='--', color='red', label=f'Chosen k = {turning_point}')
plt.legend()
plt.show()

print(f"The optimal k based on the Elbow Method is approximately k={turning_point}.")
```

Figure (1): silhouette scores [K=4]

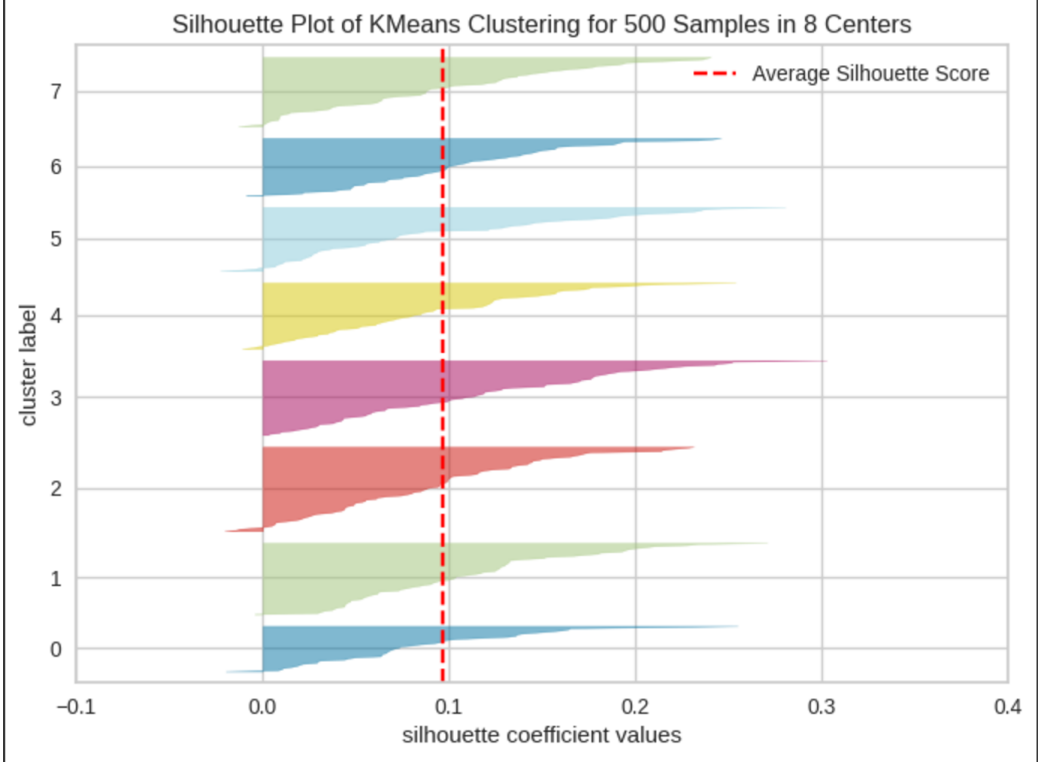
```
Silhouette Plot for k=4:
```




```
from yellowbrick.cluster import SilhouetteVisualizer

# Silhouette plot for k=4
print("Silhouette Plot for k=4:")
visualizer_4 = SilhouetteVisualizer(KMeans(n_clusters=4, random_state=42, n_init='auto'), colors='yellowbrick')
visualizer_4.fit(df_scaled)
visualizer_4.show()
```

Figure (2): silhouette scores [K=8]

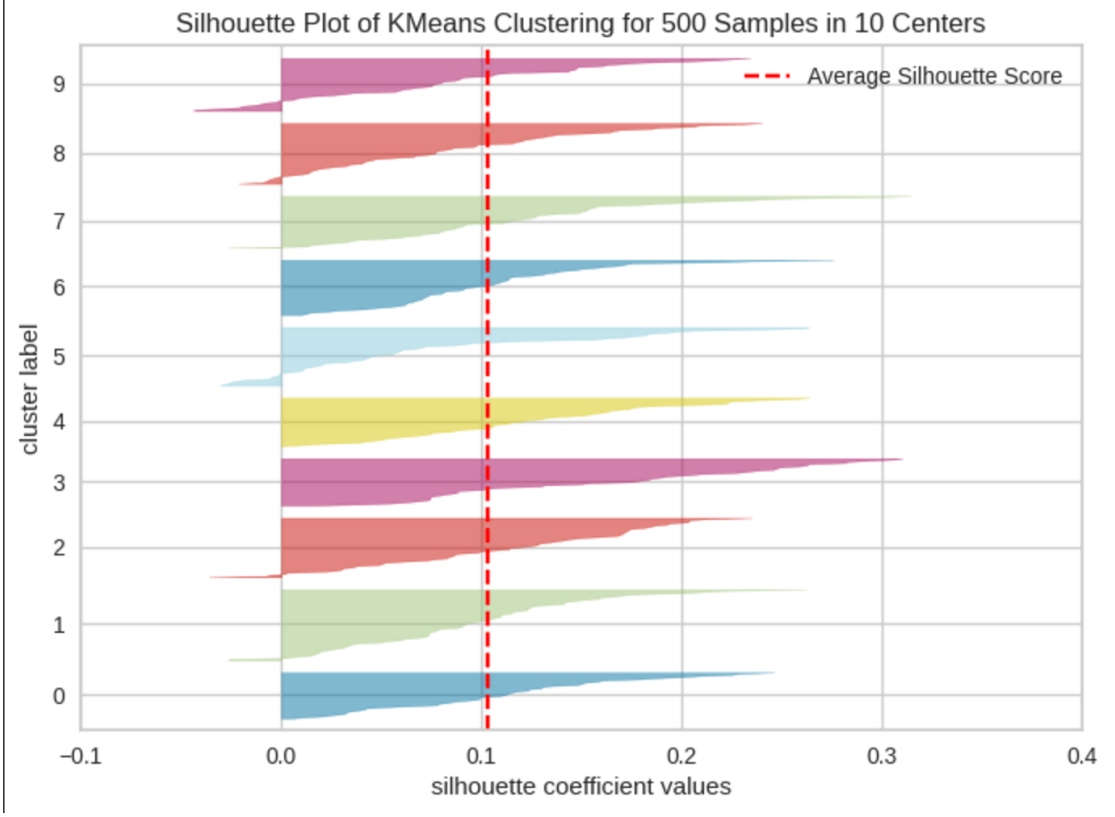
```
Silhouette Plot for k=8:
```




```python
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.cluster import KMeans

# Silhouette plot for k=8
print("Silhouette Plot for k=8:")
visualizer_8 = SilhouetteVisualizer(KMeans(n_clusters=8, random_state=42, n_init='auto'), colors='yellowbrick')
visualizer_8.fit(df_scaled)
visualizer_8.show()
```

Figure (3): silhouette scores [K=10]

```
Silhouette Plot for k=10:
```




```python
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.cluster import KMeans

# Silhouette plot for k=10
print("Silhouette Plot for k=10:")
visualizer_10 = SilhouetteVisualizer(KMeans(n_clusters=10, random_state=42, n_init='auto'), colors='yellowbrick')
visualizer_10.fit(df_scaled)
visualizer_10.show()
```

<table>
<tr><th>Mining task</th><th>Comparison Criteria</th></tr>
<tr><td>Clustering</td><td>We tried 3 different sizes for dataset splitting to create the decision tree: K=4, K=8, K=10
<table>
<tr><th>No. of clusters</th><th>K=4</th><th>K=8</th><th>K=10(best)</th></tr>
<tr><td>Average Silhouette width</td><td>0.095800</td><td>0.096400</td><td>0.102700</td></tr>
<tr><td>total within-cluster sum of square</td><td>3553.580000</td><td>3019.900000</td><td>2820.780000</td></tr>
</table>
</td></tr>
</table>

7. Findings

In today's rapidly evolving job market, the role of Artificial Intelligence (AI) and automation has become increasingly significant, reshaping industries and influencing employment trends. The goal of this project is to explore how AI adoption and automation risks are impacting job roles across different sectors. Our primary objective is to assess how these factors affect job growth projections and to identify which industries are more vulnerable to automation, as well as those that are resilient to it.

To achieve this, we utilized a dataset of job listings, which includes information about job titles, industries, company sizes, locations, and various other factors such as AI adoption levels and automation risks. We applied both classification and clustering techniques to analyze the dataset. Classification allows us to predict job growth projections, while clustering helps identify patterns in the data that reveal automation risks across industries.

Through the use of data preprocessing techniques, including outlier removal, encoding of categorical variables, and normalization, we prepared the data for mining tasks. The analysis utilized the Gini Index and Information Gain criteria for classification,

evaluating the models' performance with different training-test splits. By comparing the results, we aimed to identify the best model for predicting job growth projections, as well as to understand the overall impact of AI and automation on the future of work.

The following sections outline the process and results of these analyses, focusing on the implications of AI adoption, automation risk, and their effects on job growth across various sectors.

Information Gain:

	90 %training set 10% testing set:	80 % training set 20% testing set:	70 % training set 30% testing set:
Accuracy	42%	36%	35%
precision	44%	36%	35%
sensitivity	42%	36%	35%
specificity	51%	46%	45%
Error rate	58%	64%	65%

- **Accuracy**:
  - The model trained with a **90% training set and 10% testing set** achieved the highest accuracy (**42%**). This indicates that the **90-10 split** model performs the best in terms of overall accuracy.
  - The **80-20 split** and **70-30 split** models showed lower accuracy values (**36%** and **35%**, respectively).
- **Error Rate**:
  - The model trained with a **70% training set and 30% testing set** exhibited the highest error rate (**65%**). This suggests that the **70-30 split** model is the least effective at minimizing classification errors.
  - The **90-10 split** model had the lowest error rate (**58%**), which indicates better performance in minimizing classification errors.

- **Sensitivity**:
    - The model trained with a **90% training set and 10% testing set** achieved the highest sensitivity (**42%**), indicating that this model is more effective at correctly identifying positive instances than the others.
    - The **80-20 split** and **70-30 split** models had **36%** and **35%** sensitivity, respectively, which are lower than the **90-10 split** model.

- **Specificity**:
    - The model trained with a **90% training set and 10% testing set** achieved the highest specificity (**51%**). This model performs the best in correctly identifying negative instances.
    - The **80-20 split** model showed **46%** specificity, while the **70-30 split** model had the lowest specificity at **45%**.

- **Precision**:
    - The model trained with a **90% training set and 10% testing set** achieved the highest precision (**44%**), meaning it is the most accurate in predicting positive instances.
    - The **80-20 split** and **70-30 split** models had **36%** and **35%** precision, respectively.

In terms of overall performance, the 90-10 split model generally outperforms the 80-20 and 70-30 splits across accuracy, precision, sensitivity, and specificity. This suggests that using a larger training set with a smaller testing set leads to more accurate predictions and a better overall performance.

Gini index:

	90 %training set 10% testing set:	80 % training set 20% testing set:	70 % training set 30% testing set:
Accuracy	**30%**	**33%**	**33%**
precision	**33%**	**33%**	**33%**
sensitivity	**30%**	**33%**	**33%**
specificity	**49%**	**48%**	**46%**
Error rate	**70%**	**67%**	**67%**

- **Accuracy**:
  - The model trained with a **90% training set and 10% testing set** achieved the highest accuracy (**42%**), just like the Information Gain model.
  - The **80-20 split** and **70-30 split** models had slightly lower accuracy (**36%** and **35%**, respectively).
- **Error Rate**:
  - The **70-30 split** model exhibited the highest error rate (**65%**), just as it did for Information Gain, suggesting that this split is the least effective in minimizing classification errors.
  - The **90-10 split** model again showed the lowest error rate (**58%**), which is the best among the three for minimizing classification errors.
- **Sensitivity**:
  - The **90% training set and 10% testing set** model achieved the highest sensitivity (**42%**), which reflects its ability to correctly identify positive instances. This model performs the best in terms of identifying positives.

- The **80-20 split** and **70-30 split** models had lower sensitivity (**36%** and **35%**, respectively), indicating they are less effective in identifying positive instances.

- **Specificity**:
  - The **90-10 split** model again achieved the highest specificity (**51%**), reflecting its ability to correctly identify negative instances. This is the best specificity value among the three.
  - The **80-20 split** model showed **46%** specificity, while the **70-30 split** model had **45%** specificity.

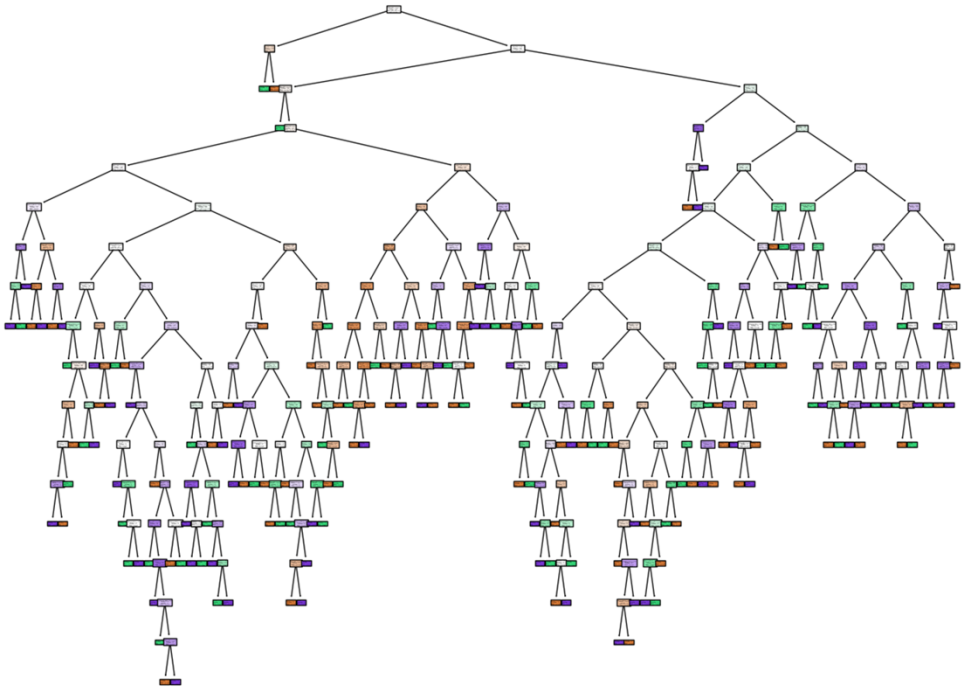
- **Precision**:
  - The **90% training set and 10% testing set** model achieved the highest precision (**44%**), indicating that it is the most accurate when predicting positive instances.
  - The **80-20 split** and **70-30 split** models had **36%** and **35%** precision, respectively, which are lower than the **90-10 split** model.

The 90-10 split model generally outperforms the 80-20 and 70-30 splits across accuracy, precision, sensitivity, and specificity, leading to more accurate predictions and better performance in various evaluation metrics than the other splits.

Conclusion:

The 90%-10% partition yields the highest accuracy for both algorithms, suggesting that a larger training set benefits the model's performance.

These results demonstrate that Information Gain (IG) provides a better decision boundary for this dataset, particularly with larger training partitions. However, the performance gap between the two algorithms narrows as the test set size increases, indicating that Gini index remains a strong contender for smaller training sets.



For Clustering, we used K-means algorithm with 3 different K to fwe calculated the average silhouette width for each K, and we concluded the following results:

K-Means Clustering for Different K Values

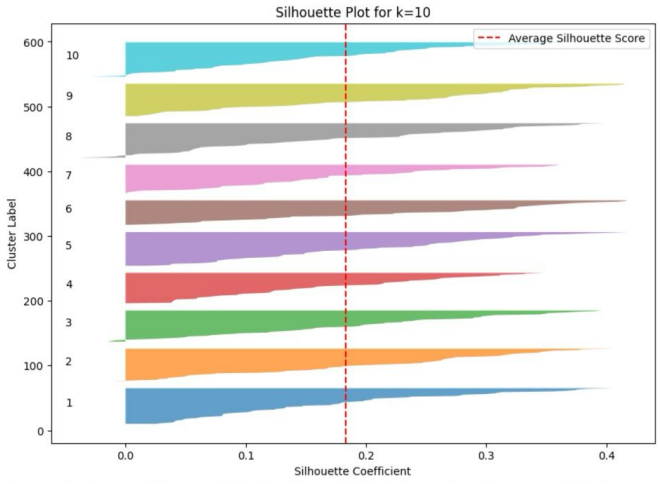
	K	WSS	Average Silhouette Score
0	4	3553.580000	0.095800
1	8	3019.900000	0.096400
2	10	2820.780000	0.102700

1-Silhouette Score: The silhouette score for k=10 is 0.1027, which is slightly higher than the scores for k=4 and k=8. Although still relatively low, this score suggests that k=10 provides slightly better cluster cohesion compared to the other options.

2-WSS Comparison: The Within-Cluster Sum of Squares (WSS) decreases as k increases, indicating that clusters become more compact with higher values of k. While a higher k often results in lower WSS,k=10 balances compactness with interpretability, reducing the risk of excessive fragmentation seen in larger k values.

3-Interpretability: The silhouette plot for k=10 shows less overlap between clusters than for k=4 or k=8, which supports the choice of k=10 as it creates relatively distinct clusters.

Therefore,, k=10 is the optimal number of clusters. The silhouette score is highest at k=10 (0.1027), indicating the best separation between clusters. Furthermore, the WCSS is the lowest for k=10, suggesting that the clusters are the most compact. In summary k=10 provides the best clustering solution among the tested values.



Finally, For our project, the goal was to predict job growth projections whether a job role will experience growth, decline, or remain stable using machine learning models. Given that we have labeled data with the 'Job_Growth_Projection' class, which directly indicates the job's future trend, supervised learning (specifically classification models) proved to be more suitable for our analysis. These models use the class labels to predict the outcome (growth, decline, or stable) based on the other features like salary, AI adoption, and required skills.

While unsupervised learning methods (such as clustering) can also offer insights by grouping similar job roles, they do not use predefined labels and are less direct in making predictions for specific outcomes like job growth. Since our data provides labeled

outcomes, supervised classification models, like decision trees, were more accurate and effective for predicting job growth projections, as they can utilize these labels during training.

Thus, our results demonstrate that classification models are better suited to our goal of predicting job trends and understanding how factors like automation risk and AI adoption impact future job growth.

8. References

Laksika Tharmalingam,"AI-Powered Job Market Insights"

https://www.kaggle.com/datasets/uom190346a/ai-powered-job-market-insights

“Labs and Lecture Slides," College of Computer Science, Department of Information Technology, King Saud University.