Introduction

This project utilizes the k-Nearest Neighbors (KNN) algorithm to classify companies into "Tech" and "Non-Tech" categories based on the number of employees and revenue. The dataset is generated hypothetically, and features are standardized for model training. The trained KNN model is then employed to predict the company types on a test set. The project concludes with a comprehensive evaluation, including accuracy, a classification report, and a confusion matrix visualization.

Description

1. Dataset Generation:

In this step, we generated a hypothetical dataset for companies. We simulated data for the number of employees and revenue, and based on certain conditions, we classified companies into two types: "Tech" and "Non-Tech." This dataset creation is crucial for training and testing our KNN model.

2. Data Exploration and Visualization:

We used Seaborn and Matplotlib to visualize the dataset through a scatterplot. The x-axis represents the number of employees, the y-axis represents revenue, and points are colored based on company type (Tech or Non-Tech). This visualization allows us to understand the distribution of companies in the dataset.

3. Data Splitting and Standardization:

The dataset was split into features (X: number of employees and revenue) and labels (y: company type). 80% of the data was designated for training, and 20% for testing using train_test_split. Features were standardized using StandardScaler to ensure consistency in their contributions to the KNN algorithm.

4. KNN Model Creation and Training:

A KNN classifier was created with n_neighbors=3, indicating it considers the three nearest neighbors during predictions. The model was trained using the standardized training data (X_train_scaled and y_train). The KNN algorithm learns the patterns present in the training data.

5. Making Predictions:

The trained KNN model was employed to make predictions on the test set (X_test_scaled). Each prediction represents the type of company (Tech or Non-Tech) based on the features provided.

6. Model Evaluation:

- **Accuracy Calculation:**

The accuracy of the model was calculated using the accuracy_score function. It represents the ratio of correctly predicted instances to the total number of instances in the test set.

- **Classification Report:**

A detailed classification report was generated using the classification_report function. It includes metrics such as precision, recall, and F1-score for both "Tech" and "Non-Tech" categories, providing a comprehensive understanding of the model's performance.

- **Confusion Matrix Visualization:**

A confusion matrix was created using the confusion_matrix function. This matrix visually represents the count of true positives, true negatives, false positives, and false negatives. A heatmap of the confusion matrix was plotted using Seaborn, aiding in the interpretation of the model's classification results.

Outcomes

- The scatterplot visualization provided insights into the distribution of companies in the dataset, highlighting patterns that may influence the classification.
- The KNN model achieved a certain accuracy on the test set, providing an overall measure of its predictive performance.
- The classification report detailed precision, recall, and F1-score for each category, offering a nuanced assessment of the model's ability to classify both "Tech" and "Non-Tech" companies.
- The confusion matrix and its heatmap provided a visual representation of the model's performance, illustrating areas of correct and incorrect predictions.

Code

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns



# Generate a hypothetical companies dataset

np.random.seed(42)
```

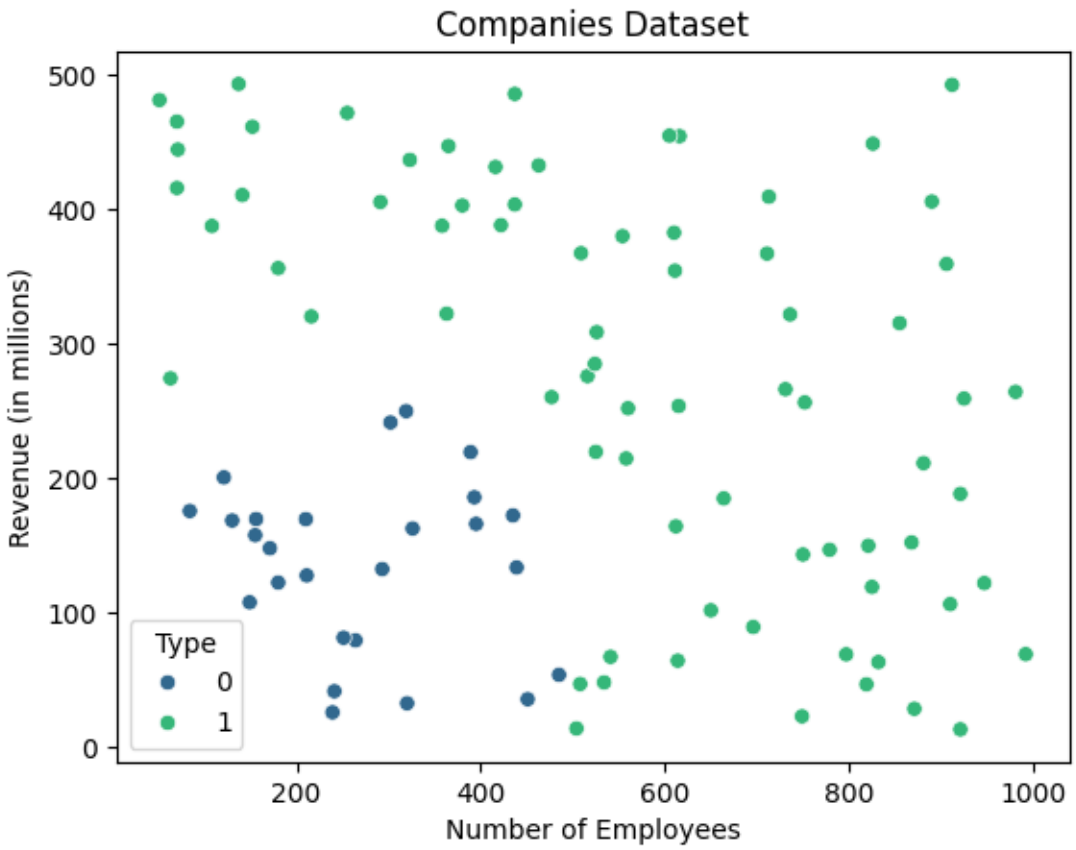
```
# Features: Number of Employees and Revenue (in millions)
data = {
    'Number_of_Employees': np.random.randint(50, 1000, 100),
    'Revenue': np.random.uniform(10, 500, 100),
}

# Target: Tech (1) or Non-Tech (0)
data['Type'] = np.where((data['Number_of_Employees'] > 500) | (data['Revenue'] > 250), 1, 0)

companies_df = pd.DataFrame(data)

# Explore the data
sns.scatterplot(data=companies_df, x='Number_of_Employees', y='Revenue', hue='Type',
palette='viridis')
plt.title('Companies Dataset')
plt.xlabel('Number of Employees')
plt.ylabel('Revenue (in millions)')
plt.show()

# Separate features (X) and labels (y)
X = companies_df[['Number_of_Employees', 'Revenue']]
```

```
y = companies_df['Type']


# 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)


# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


# Create a KNN classifier
knn_classifier = KNeighborsClassifier(n_neighbors=3)


# Train the model
knn_classifier.fit(X_train_scaled, y_train)


# Makes predictions on the test set
y_pred = knn_classifier.predict(X_test_scaled)


# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f'Accuracy on the test set: {accuracy:.2f}')



print('Classification Report:')

print(classification_report(y_test, y_pred))



# Confusion Matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', xticklabels=knn_classifier.classes_,
yticklabels=knn_classifier.classes_)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()
```

Output

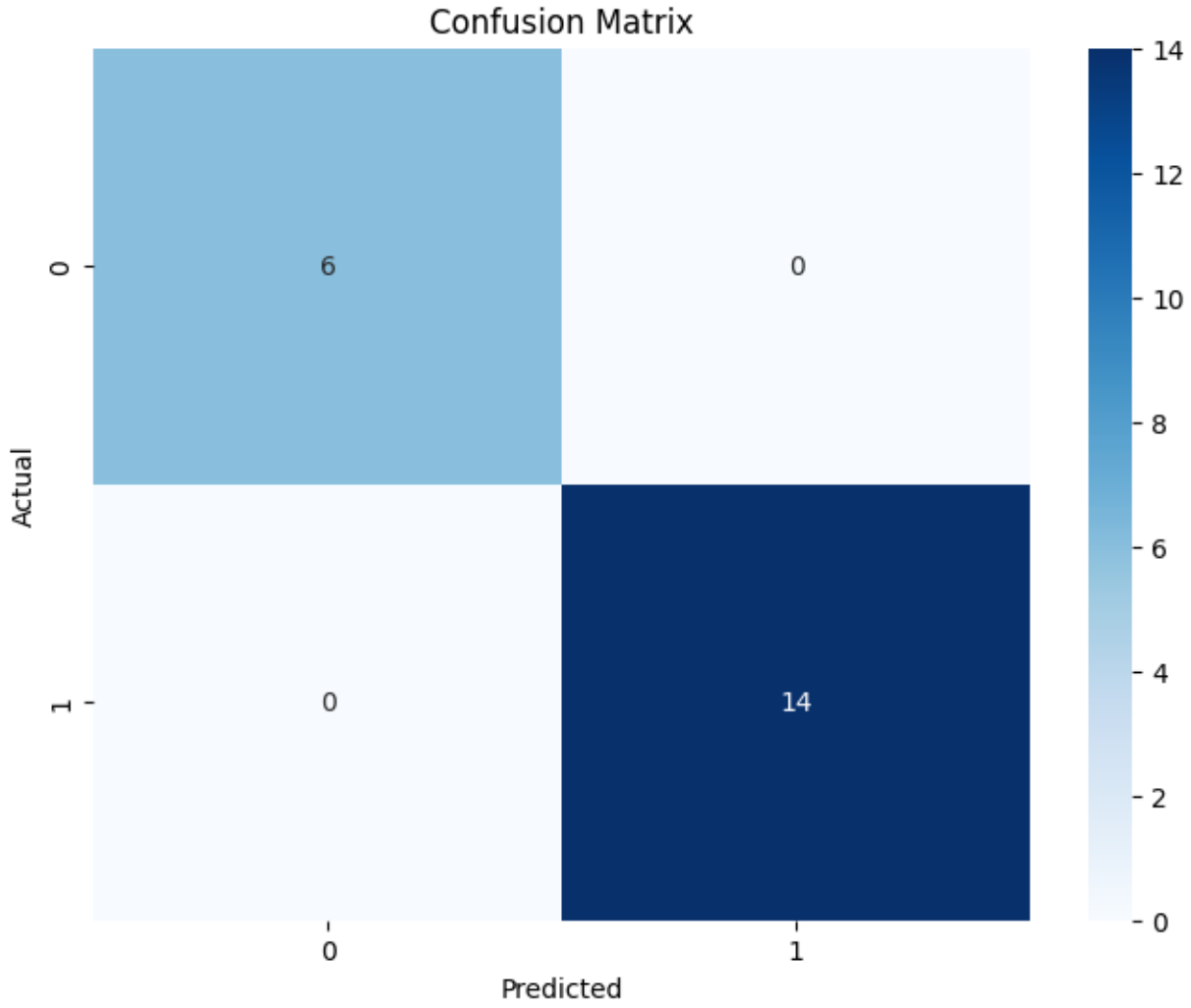
Scatter Plot (Visualization):



Accuracy and Classification Report:

```
Accuracy on the test set: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00        14

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Confusion Matrix (Heat Map):



Conclusion

This project provided a hands-on exploration of machine learning and artificial intelligence concepts using the k-Nearest Neighbors (KNN) algorithm to classify companies based on simulated features. Key learnings include the importance of realistic dataset generation, visual exploration for pattern recognition, and the significance of data preprocessing steps such as splitting and standardization. The practical implementation of KNN deepened our understanding of how proximity-based algorithms operate. Additionally, the model evaluation process, encompassing metrics like accuracy, precision, recall, and the confusion matrix, illuminated the strengths and areas for improvement in the classification model. These insights form a solid foundation for future machine learning and artificial intelligence projects, emphasizing the iterative process of data understanding, model development, and performance evaluation in artificial intelligence.