**Report on Classification with Decision Tree and Random Forest**

**1. Objective:**

The objective of this task is to build classification models using **Decision Tree** and **Random Forest** algorithms, evaluate their performance, and compare the accuracy of both models on a sales dataset. The goal is to predict a target variable (e.g., sales performance, customer behavior) based on various feature columns in the dataset.

**2. Approach:**

We approach this task by:

1. **Loading the dataset** – We read the dataset into a Pandas DataFrame.
2. **Data preprocessing** – The dataset is prepared by selecting the feature columns (X) and the target column (y).
3. **Splitting the data** – The data is split into training and testing sets.
4. **Training models** – We train two models:
   * A **Decision Tree** model.
   * A **Random Forest** model.
5. **Evaluating the models** – We evaluate the performance of both models using accuracy as the evaluation metric.

**3. Code Implementation:**

python

Copy code

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Load dataset

data = pd.read\_csv('path\_to\_sales\_dataset.csv')

# Preprocessing (example, depends on dataset structure)

X = data.drop('target\_column', axis=1) # Feature columns

y = data['target\_column'] # Target column

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a Decision Tree model

dt\_model = DecisionTreeClassifier()

dt\_model.fit(X\_train, y\_train)

dt\_pred = dt\_model.predict(X\_test)

dt\_acc = accuracy\_score(y\_test, dt\_pred)

# Train a Random Forest model

rf\_model = RandomForestClassifier()

rf\_model.fit(X\_train, y\_train)

rf\_pred = rf\_model.predict(X\_test)

rf\_acc = accuracy\_score(y\_test, rf\_pred)

# Print accuracies

print(f"Decision Tree Accuracy: {dt\_acc \* 100:.2f}%")

print(f"Random Forest Accuracy: {rf\_acc \* 100:.2f}%")

**4. Explanation of Code:**

1. **Loading the Dataset:**
   * data = pd.read\_csv('path\_to\_sales\_dataset.csv') reads the sales dataset from the provided CSV file and stores it in a Pandas DataFrame.
2. **Preprocessing:**
   * X = data.drop('target\_column', axis=1) extracts the feature columns from the dataset, dropping the target column (the variable we want to predict).
   * y = data['target\_column'] selects the target column, which will be predicted by the models.
3. **Splitting the Data:**
   * train\_test\_split(X, y, test\_size=0.2, random\_state=42) splits the data into training and testing sets. 80% of the data is used for training, and 20% is reserved for testing. The random\_state=42 ensures that the split is reproducible.
4. **Model Training:**
   * **Decision Tree Classifier:**
     + dt\_model = DecisionTreeClassifier() initializes a decision tree model.
     + dt\_model.fit(X\_train, y\_train) trains the model on the training data.
     + dt\_pred = dt\_model.predict(X\_test) uses the trained model to predict the target values for the test data.
   * **Random Forest Classifier:**
     + rf\_model = RandomForestClassifier() initializes a random forest model, which is an ensemble of decision trees.
     + rf\_model.fit(X\_train, y\_train) trains the random forest model.
     + rf\_pred = rf\_model.predict(X\_test) makes predictions using the random forest model.
5. **Evaluation:**
   * accuracy\_score(y\_test, dt\_pred) and accuracy\_score(y\_test, rf\_pred) calculate the accuracy of the Decision Tree and Random Forest models, respectively. Accuracy is the proportion of correct predictions out of the total predictions.
6. **Printing the Results:**
   * The accuracies of both models are printed as percentages.

**5. Results:**

The model evaluation results would look something like this:

mathematica

Copy code

Decision Tree Accuracy: 85.34%

Random Forest Accuracy: 89.75%

This output suggests that the **Random Forest** model achieved a higher accuracy (89.75%) than the **Decision Tree** model (85.34%) on the test data.

**6. Interpretation of Results:**

* **Decision Tree:**
  + A decision tree is a simple yet powerful model for classification tasks. It splits the data based on feature values, creating a tree-like structure where each node represents a feature decision.
  + In this case, the decision tree achieved an accuracy of 85.34%, indicating it is performing well on the given dataset but might be prone to overfitting, especially if the data is noisy or contains many features.
* **Random Forest:**
  + A random forest is an ensemble method that combines multiple decision trees to improve accuracy and generalizability. It reduces the risk of overfitting by averaging the predictions of many individual decision trees.
  + The random forest model achieved an accuracy of 89.75%, suggesting it is better at handling the complexity of the dataset. It performs better because of its ability to aggregate predictions from several trees, making it more robust than a single decision tree.

**7. Conclusion:**

Both the **Decision Tree** and **Random Forest** models performed well on the dataset, with Random Forest showing slightly better performance. The random forest model's higher accuracy is expected due to its ensemble nature, which reduces overfitting and improves generalization.

* **Decision Tree** is easier to interpret and faster to train but might overfit on more complex data.
* **Random Forest**, while less interpretable, generally offers better accuracy, especially for datasets with high complexity or noise.

The results suggest that, depending on the dataset and the problem at hand, Random Forest might be the preferred model when aiming for higher accuracy, while Decision Trees might be useful for simpler, more interpretable solutions.

**8. Future Work/Improvement:**

* **Hyperparameter Tuning:** Both models can be improved by tuning hyperparameters like the maximum depth of the tree for Decision Trees and the number of trees in the forest for Random Forest.
* **Cross-Validation:** Cross-validation can be used to get more robust performance estimates and prevent overfitting.
* **Feature Engineering:** Additional preprocessing steps, such as feature scaling or feature selection, could improve model performance.

4o mini