Building and Evaluating a Decision Tree Classifier on a real world dataset

Table of Contents

Description

In this lab, we will explore the **House Prices dataset** from Kaggle, which contains information about various properties in Ames, Iowa, along with their sale prices. This dataset includes numerous features such as the size of the house, the number of bedrooms and bathrooms, the condition of the property, and other relevant attributes. The goal is to utilize a **Decision Tree Classifier** to predict house prices based on these features, demonstrating the application of machine learning techniques on a real-world dataset.

Problem Statement

The objective of this lab is to develop a model that accurately predicts the sale prices of houses based on their features. By implementing a **Decision Tree Classifier**, we aim to understand how this algorithm can be

employed to handle regression tasks effectively. We will train our model on a portion of the dataset and evaluate its performance on unseen data, providing insights into its accuracy and ability to generalize.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-02) is required before proceeding with Lab Guide-03.

Software Requirements

- **Python Version**: Python 3.11.9
- **Integrated Development Environment**: Visual Studio Code (VSCode)
- **Required Libraries**:
  - `pandas`
  - `numpy`
  - `scikit-learn`
  - `matplotlib`
  - `seaborn`

Hardware Requirements

- Minimum **4 GB RAM**
- Recommended **8 GB RAM** or more for optimal performance
- **CPU**: Intel i5 or equivalent

Setup Instructions

Setting Up a Python Environment

1. Install Python: Download and install Python 3.11.9 from the **official Python website**.

- Locate a reliable version of Python 3, **"Download Python 3.11.9"**.
- Choose the correct link for your device from the options provided: either Windows installer (64-bit) or Windows installer (32-bit) and proceed to download the executable file.

2. Install Visual Studio Code: Download and install VSCode from the **official website**.

3. Install Required Extensions for VSCode

- Python extension for VSCode (search for "Python" in the Extensions Marketplace).

4. Create a Virtual Environment

- Open VSCode Terminal (`Ctrl + ` `).
- Run the following commands to create a virtual environment:

```
python -m venv house_price_env
```

- Activate the virtual environment for Windows:

```
house_price_env\Scripts\activate
```

5. Install Required Libraries

- With the virtual environment activated, run:

```
pip install pandas numpy scikit-learn matplotlib seaborn
```

```
PS C:\Users\Administrator\Desktop\AIML> pip install pandas numpy scikit-learn matplotlib seaborn
Collecting pandas
  Using cached pandas-2.2.3-cp311-cp311-win_amd64.whl.metadata (19 kB)
Collecting numpy
  Using cached numpy-2.1.2-cp311-cp311-win_amd64.whl.metadata (59 kB)
Collecting scikit-learn
  Using cached scikit_learn-1.5.2-cp311-cp311-win_amd64.whl.metadata (13 kB)
Collecting matplotlib
  Using cached matplotlib-3.9.2-cp311-cp311-win_amd64.whl.metadata (11 kB)
Collecting seaborn
  Using cached seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\program files\python311\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\program files\python311\lib\site-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in c:\program files\python311\lib\site-packages (from pandas) (2024.2)
Requirement already satisfied: scipy>=1.6.0 in c:\program files\python311\lib\site-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in c:\program files\python311\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\program files\python311\lib\site-packages (from scikit-learn) (3.5.0)
4.7)                                                                                                        .0)
Requirement already satisfied: packaging>=20.0 in c:\program files\python311\lib\site-packages (from matplotlib) (24.1
1)                                                                                                          54.1)
Requirement already satisfied: pillow>=8 in c:\program files\python311\lib\site-packages (from matplotlib) (11.0.0)   4.7)
Requirement already satisfied: pyparsing>=2.3.1 in c:\program files\python311\lib\site-packages (from matplotlib) (3.2.0)
Requirement already satisfied: six>=1.5 in c:\program files\python311\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Using cached pandas-2.2.3-cp311-cp311-win_amd64.whl (11.6 MB)
Requirement already satisfied: pyparsing>=2.3.1 in c:\program files\python311\lib\site-packages (from matplotlib) (3.2.0)
Requirement already satisfied: six>=1.5 in c:\program files\python311\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Using cached pandas-2.2.3-cp311-cp311-win_amd64.whl (11.6 MB)
Using cached numpy-2.1.2-cp311-cp311-win_amd64.whl (12.9 MB)
Using cached scikit_learn-1.5.2-cp311-cp311-win_amd64.whl (11.0 MB)
Using cached matplotlib-3.9.2-cp311-cp311-win_amd64.whl (7.8 MB)
Using cached seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: numpy, pandas, scikit-learn, matplotlib, seaborn
Successfully installed matplotlib-3.9.2 numpy-2.1.2 pandas-2.2.3 scikit-learn-1.5.2 seaborn-0.13.2
```

Key Terms

- **Decision Tree**: A flowchart-like structure used for decision-making, breaking down a dataset into smaller subsets while developing an associated decision tree incrementally.
- **Regression**: A type of predictive modeling technique that estimates the relationships among variables. In this context, it is used to predict continuous outcomes (house prices).
- **Overfitting**: A modeling error that occurs when a model captures noise instead of the underlying data distribution, leading to poor performance on unseen data.
- **Feature Importance**: A technique to quantify the contribution of each feature in making predictions, helping to interpret the model's behavior.

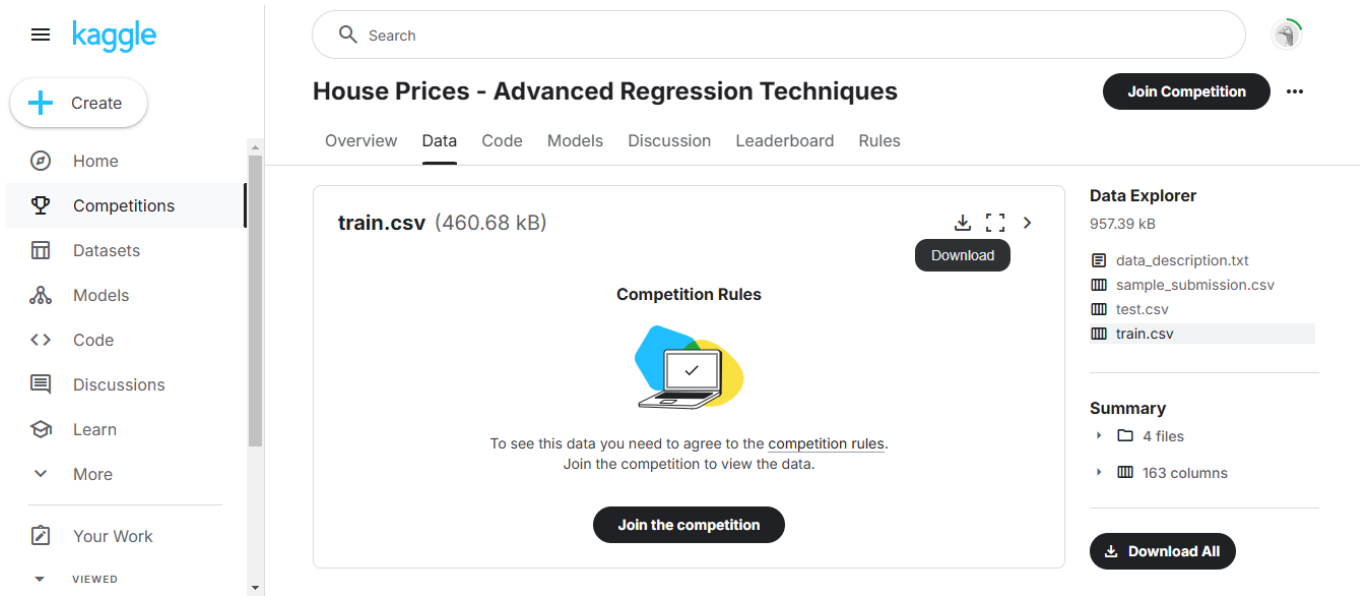
Example Usage

Downloading the Dataset

1. Sign in to Kaggle: Go to the **Kaggle website** and sign in to your account. If you don't have an account, create one.

2. Download the Dataset:

- Navigate to the **House Prices: Advanced Regression Techniques** competition page.



- Click on the "Data" tab and download the `train.csv` file (the dataset used for training).
- Move the downloaded `train.csv` file into your project directory.

Loading the Dataset

Create a new Python file (e.g., `decision_tree.py`) in VSCode and add the following code to load the dataset:

```
import pandas as pd

# Load the dataset
data = pd.read_csv('./train.csv')  # Adjust the path if necessary
print(data.head())
```

Output

```
PS C:\Users\Administrator\Desktop\AIML> python decision_tree.py
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street  ... MiscVal MoSold YrSold SaleType SaleCondition SalePrice
0   1          60       RL         65.0     8450   Pave  ...       0      2   2008       WD        Normal    208500
1   2          20       RL         80.0     9600   Pave  ...       0      5   2007       WD        Normal    181500
2   3          60       RL         68.0    11250   Pave  ...       0      9   2008       WD        Normal    223500
3   4          70       RL         60.0     9550   Pave  ...       0      2   2006       WD       Abnorml    140000
4   5          60       RL         84.0    14260   Pave  ...       0     12   2008       WD        Normal    250000

[5 rows x 81 columns]
```

Data Preprocessing

Before training the model, we need to preprocess the dataset by handling missing values and encoding categorical variables. Add the following code for preprocessing:

```
# Handle missing values
# Fill missing values for numeric columns with their mean
```

```
numeric_cols = data.select_dtypes(include=['float64', 'int64']).columns  # Get
numeric columns
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].mean())  #
Replace NaN with mean

# Fill missing values for categorical columns with their mode
categorical_cols = data.select_dtypes(include=['object']).columns  # Get
categorical columns
for col in categorical_cols:
    data[col].fillna(data[col].mode()[0], inplace=True)  # Fill with mode (most
common value)

# Convert categorical variables to dummy/indicator variables
data = pd.get_dummies(data, drop_first=True)

# Define features and target variable
X = data.drop(['SalePrice'], axis=1)  # Features
y = data['SalePrice']  # Target variable
```

Output

```
PS C:\Users\Administrator\Desktop\AIML> python decision_tree.py
C:\Users\Administrator\Desktop\AIML\decision_tree.py:15: FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are sett
ing values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df
[col].method(value) instead, to perform the operation inplace on the original object.


  data[col].fillna(data[col].mode()[0], inplace=True) # Fill with mode (most
```

Implementing the Decision Tree Classifier

Add the following code to implement the Decision Tree Classifier:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor

# Splitting the dataset 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)

# Creating and training the model
model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)
```

Output

```
PS C:\Users\Administrator\Desktop\AIML> python decision_tree.py
DecisionTreeRegressor(random_state=42)
```

Evaluating the Model

Add the following code to evaluate the model:

```
from sklearn.metrics import mean_squared_error

# Making predictions
predictions = model.predict(X_test)

# Evaluating the model
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse:.2f}')
```

Output

```
Mean Squared Error: 1682563175.07
```

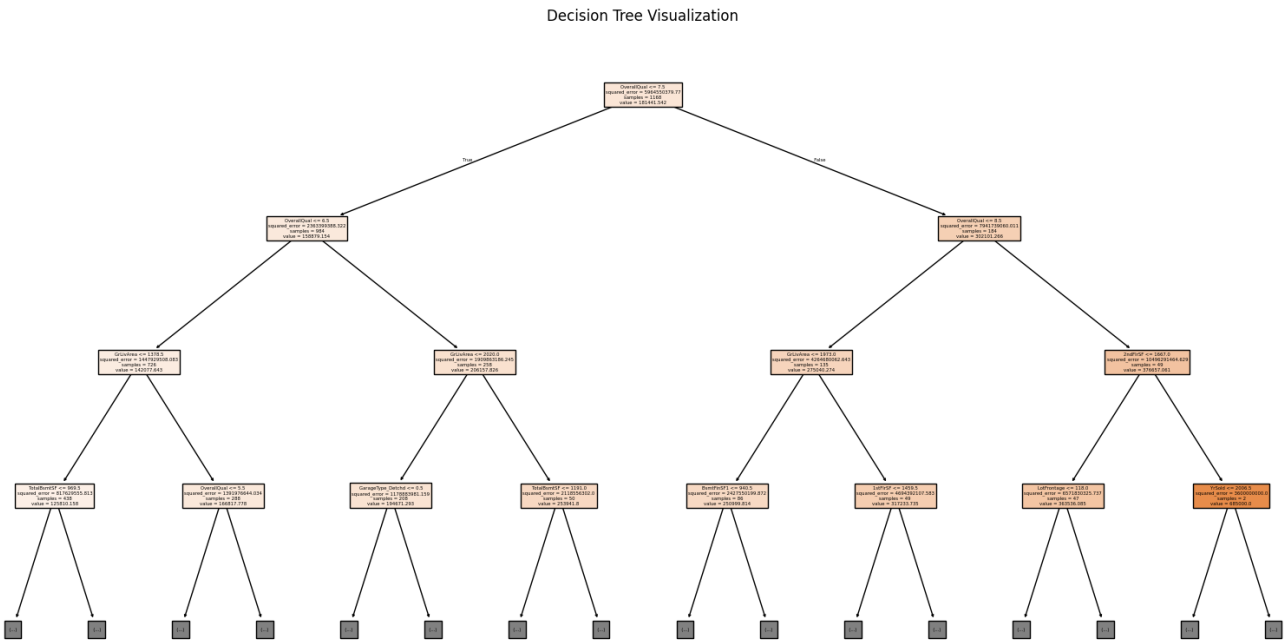
Visualizing the Decision Tree

To visualize the decision tree, add the following code:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))
plot_tree(model, filled=True, feature_names=X.columns, max_depth=3)
plt.title('Decision Tree Visualization')
plt.show()
```

Output



Visualizing Feature Importances

Finally, to visualize feature importances, add the following code:

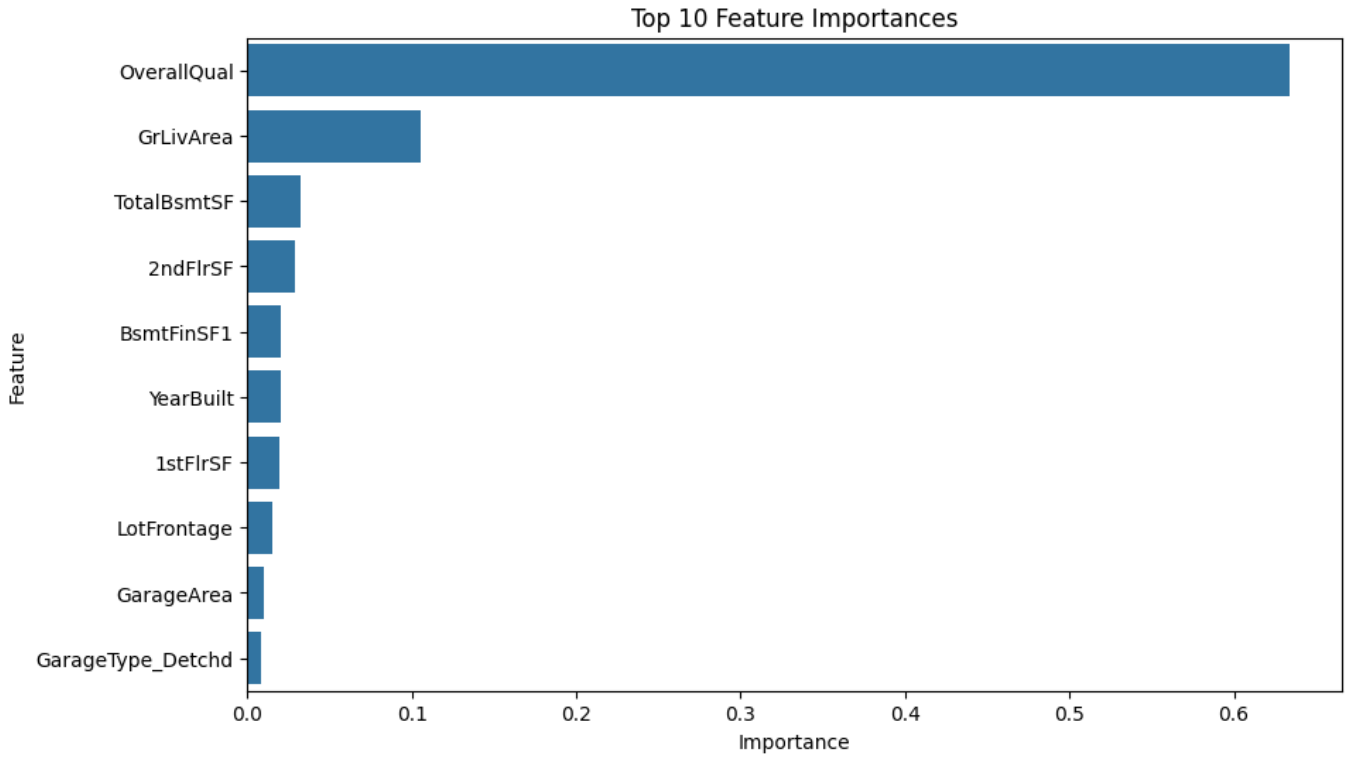
```
import seaborn as sns

# Getting feature importances
importances = model.feature_importances_

# Creating a DataFrame for visualization
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance':
importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance',
ascending=False)

# Visualizing feature importances
plt.figure(figsize=(10,6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df.head(10))
plt.title('Top 10 Feature Importances')
plt.show()
```

Output



References

- Kaggle House Prices Dataset
- Matplotlib Documentation
- Decision Tree Classification