Regression_Trees_Taxi_Tip

October 13, 2025

1 Regression Trees

Estimated time needed: **30** minutes

In this exercise session you will use a real dataset to train a regression tree model. The dataset includes information about taxi tip and was collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP). You will use the trained model to predict the amount of tip paid.

1.1 Objectives

After completing this lab you will be able to:

- Perform basic data preprocessing using Scikit-Learn
- Model a regression task using Scikit-Learn
- Train a Decision Tree Regressor model
- Run inference and assess the quality of the trained models

Introduction

The dataset used in this exercise session is a subset of the publicly available <href='https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page'>TLC Dataset (all rights reserved by Taxi & Limousine Commission (TLC), City of New York). The prediction of the tip amount can be modeled as a regression problem. To train the model you can use part of the input dataset and the remaining data can be used to assess the quality of the trained model.

```
<h2>Import Libraries</h2>
```

Make sure the libraries required are available by executing the cell below.

[1]:
```
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install scikit-learn
```

```
Requirement already satisfied: numpy in /opt/conda/lib/python3.12/site-packages
(2.2.0)
Requirement already satisfied: pandas in /opt/conda/lib/python3.12/site-packages
(2.2.3)
Requirement already satisfied: numpy>=1.26.0 in /opt/conda/lib/python3.12/site-
packages (from pandas) (2.2.0)
```

```
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-
packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-
packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.12/site-
packages (3.9.3)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (4.60.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (2.2.0)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-
packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.12/site-
packages (1.6.0)
Requirement already satisfied: numpy>=1.19.5 in /opt/conda/lib/python3.12/site-
packages (from scikit-learn) (2.2.0)
Requirement already satisfied: scipy>=1.6.0 in /opt/conda/lib/python3.12/site-
packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in /opt/conda/lib/python3.12/site-
packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/opt/conda/lib/python3.12/site-packages (from scikit-learn) (3.6.0)
```

Import the libraries we need to use in this lab

[2]:
```
from __future__ import print_function
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.metrics import mean_squared_error

import warnings
warnings.filterwarnings('ignore')
```

<h2>Dataset Analysis</h2>

In this section you will read the dataset in a Pandas dataframe and visualize its content. You will also look at some data statistics.

Note: A Pandas dataframe is a two-dimensional, size-mutable, potentially heterogeneous tabular data structure. For more information: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html.

[4]:
```
# read the input data
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
 ↪pu9kbeSaAtRZ7RxdJKX9_A/yellow-tripdata.csv'
raw_data = pd.read_csv(url)
raw_data
```

[4]:

	VendorID	passenger_count	trip_distance	RatecodeID	\
0	2	1	17.63	2	
1	2	1	19.52	2	
2	2	1	17.81	2	
3	2	2	19.30	2	
4	2	1	18.75	2	
…	…	…	…	…	
41197	2	1	16.94	2	
41198	2	4	19.83	2	
41199	2	1	17.31	2	
41200	2	1	17.28	2	
41201	2	1	16.82	2	

	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	\
0	1	132	164	1	
1	1	132	236	1	
2	1	132	48	1	
3	1	132	148	1	
4	1	132	234	1	
…	…	…	…	…	
41197	1	132	164	1	
41198	1	132	166	1	
41199	1	132	137	1	
41200	1	132	233	1	
41201	1	132	170	1	

	fare_amount	mta_tax	tolls_amount	improvement_surcharge	tip_amount
0	70.0	0.5	6.94	1	16.54
1	70.0	0.5	6.94	1	16.19
2	70.0	0.5	6.94	1	12.00
3	70.0	0.5	0.00	1	5.00
4	70.0	0.5	6.94	1	10.00
…	…	…	…	…	…
41197	70.0	0.5	6.94	1	5.00
41198	70.0	0.5	6.94	1	8.00
41199	70.0	0.5	6.94	1	8.00
41200	70.0	0.5	6.94	1	16.19
41201	70.0	0.5	6.94	1	4.13

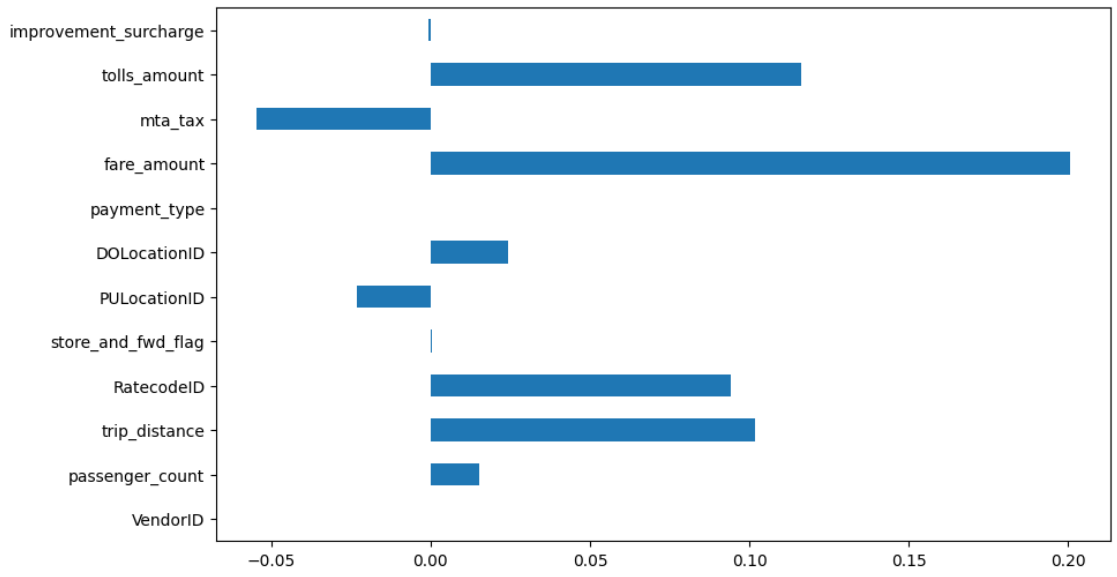
[41202 rows x 13 columns]

Each row in the dataset represents a taxi trip. As shown above, each row has 13 variables. One of the variables is `tip_amount` which will be the target variable. Your objective will be to train a model that uses the other variables to predict the value of the `tip_amount` variable.

To understand the dataset a little better, let us plot the correlation of the target variable against the input variables.

[5]:
```
correlation_values = raw_data.corr()['tip_amount'].drop('tip_amount')
correlation_values.plot(kind='barh', figsize=(10, 6))
```

[5]: <Axes: >



This shows us that the input features `payment_type`, `VendorID`, `store_and_fwd_flag` and

`improvement_surcharge` have little to no correlation with the target variable.

<h2>Dataset Preprocessing</h2>

You will now prepare the data for training by applying normalization to the input features.

[7]:
```
# extract the labels from the dataframe
y = raw_data[['tip_amount']].values.astype('float32')

# drop the target variable from the feature matrix
proc_data = raw_data.drop(['tip_amount'], axis=1)

# get the feature matrix used for training
X = proc_data.values

# normalize the feature matrix
X = normalize(X, axis=1, norm='l1', copy=False)
```

<h2>Dataset Train/Test Split</h2>

Now that the dataset is ready for building the classification models, you need to first divide the pre-processed dataset into a subset to be used for training the model (the train set) and a subset to be used for evaluating the quality of the model (the test set).

[8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

<h2>Build a Decision Tree Regressor model with Scikit-Learn</h2>

Regression Trees are implemented using `DecisionTreeRegressor`.

The important parameters of the model are:

`criterion`: The function used to measure error, we use 'squared_error'.

`max_depth` - The maximum depth the tree is allowed to take; we use 8.

[9]:
```
# import the Decision Tree Regression Model from scikit-learn
from sklearn.tree import DecisionTreeRegressor

# for reproducible output across multiple function calls, set random_state to a␣
↪given integer value
dt_reg = DecisionTreeRegressor(criterion = 'squared_error',
                               max_depth=8,
                               random_state=35)
```

Now lets train our model using the `fit` method on the `DecisionTreeRegressor` object providing our training data

[10]:
```
dt_reg.fit(X_train, y_train)
```

[10]:
```
DecisionTreeRegressor(max_depth=8, random_state=35)
```

<h2>Evaluate the Scikit-Learn and Snap ML Decision Tree Regressor Models</h2>

To evaluate our dataset we will use the `score` method of the `DecisionTreeRegressor` object providing our testing data, this number is the R^2 value which indicates the coefficient of determination. We will also evaluate the Mean Squared Error (MSE) of the regression output with respect to the test set target values. High R^2 and low MSE values are expected from a good regression model.

[11]:
```
# run inference using the sklearn model
y_pred = dt_reg.predict(X_test)

# evaluate mean squared error on the test dataset
mse_score = mean_squared_error(y_test, y_pred)
print('MSE score : {0:.3f}'.format(mse_score))

r2_score = dt_reg.score(X_test,y_test)
print('R^2 score : {0:.3f}'.format(r2_score))
```

```
MSE score : 24.555
R^2 score : 0.028
```

1.2 Practice

Q1. What if we change the max_depth to 12? How would the MSE and R^2 be affected?

Click here for the solution

MSE is noted to be increased by increasing the max_depth of the tree. This may be because of the model having excessive parameters due to which it overfits to the training data, making the performance on the testing data poorer. Another important observation would be that the model gives a negative value of R^2. This again indicates that the prediction model created does a very poor job of predicting the values on a test set.

Q2. Identify the top 3 features with the most effect on the `tip_amount`.

[13]:
```
# your code here
correlation_values = raw_data.corr()['tip_amount'].drop('tip_amount')
abs(correlation_values).sort_values(ascending=False)[:3]
```

[13]:
```
fare_amount      0.200638
tolls_amount     0.116172
trip_distance    0.101819
Name: tip_amount, dtype: float64
```

Click here for the solution

```
correlation_values = raw_data.corr()['tip_amount'].drop('tip_amount')
abs(correlation_values).sort_values(ascending=False)[:3]
```

As is evident from the output, Fare amount, toll amount and trip distance are the top features affecting the tip amount, which make logical sense.

Q3. Since we identified 4 features which are not correlated with the target variable, try removing these variables from the input set and see the effect on the MSE and R^2 value.

[14]:
```
# your code here
raw_data = raw_data.drop(['payment_type', 'VendorID', 'store_and_fwd_flag',␣
↪'improvement_surcharge'], axis=1)
```

Click here for the solution

```
raw_data = raw_data.drop(['payment_type', 'VendorID', 'store_and_fwd_flag', 'improvement_surcha
```

```
# Execute all the cells of the lab after modifying the raw data.
```

The MSE and R^2 values does not change significantly, showing that there is minimal affect of these parameters on the final regression output.

Q4. Check the effect of **decreasing** the `max_depth` parameter to 4 on the MSE and R^2 values.

Click here for the solution

You will note that the MSE value decreases and R^2 value increases, meaning that the choice of `max_depth=4` may be more suited for this dataset.

1.2.1 Congratulations! You're ready to move on to your next lesson!

1.3 Author

Abhishek Gagneja

1.3.1 Other Contributors

Jeff Grossman

<!– ## Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2024-10-31	3.0	Abhishek Gagneja	Rewrite
2020-11-03	2.1	Lakshmi	Made changes in URL
2020-11-03	2.1	Lakshmi	Made changes in URL
2020-08-27	2.0	Lavanya	Moved lab to course repo in GitLab