Mean Squared Error: 85391.21295974543

R-squared score: 0.7849013737035642

Best features with as many outliers removed

Mean Squared Error: 84937.85807123013

R-squared score: 0.7860433649034126

Best features with as many outliers removed as cyclical data.

Ann

|  |  |  |  |
| --- | --- | --- | --- |
|  | Epochs = 10 |  |  |
|  | Epochs = 15 |  |  |
|  | Epochs = 20 |  |  |
|  |  |  |  |
|  |  |  |  |

K fold

Random forest regression

Fold 1

MSE: 89160.89

R-squared: 0.78

Fold 2

MSE: 86566.37

R-squared: 0.78

Fold 3

MSE: 89230.46

R-squared: 0.78

Fold 4

MSE: 89165.59

R-squared: 0.77

Fold 5

MSE: 87486.32

R-squared: 0.78

Fold 6

MSE: 88187.00

R-squared: 0.78

Fold 7

MSE: 87652.33

R-squared: 0.78

Fold 8

MSE: 87469.28

R-squared: 0.78

Fold 9

MSE: 86541.10

R-squared: 0.78

Fold 10

MSE: 87735.91

R-squared: 0.78

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Regression model | R2 | MSE | RMSE |  |
| Random Forest Regressor | 0.777446085954009 | 88350.86023811113 |  |  |
| Decision Tree Regressor | 0.7742179293852446 | 89632.39424777453 |  |  |
| Ada boost regressor | 0.7112395043766521 | 114633.96768584345 |  |  |
| Gradient boosting regressor | 0.7784494633344445 | 87952.53314022586 |  |  |
| K neighbours regressor | 0.7376374422798482 | 104154.34735538432 |  |  |

Decision tree k fold

Chart, box and whisker chart

Description automatically generated

Fold 1

MSE: 90067.46

R-squared: 0.77

Fold 2

MSE: 87682.65

R-squared: 0.78

Fold 3

MSE: 90327.81

R-squared: 0.78

Fold 4

MSE: 90051.23

R-squared: 0.77

Fold 5

MSE: 88547.55

R-squared: 0.78

Fold 6

MSE: 89384.54

R-squared: 0.77

Fold 7

MSE: 88573.64

R-squared: 0.78

Fold 8

MSE: 88497.81

R-squared: 0.77

Fold 9

MSE: 87574.12

R-squared: 0.77

Fold 10

MSE: 88773.88

R-squared: 0.77

Decision tree performs significantly faster than random forest and other models, but results a slightly lower r2 score, as shown by the boxplot

K fold sequential ann model

Chart, box and whisker chart

Description automatically generated

Epoch 1/4

34897/34897 [==============================] - 76s 2ms/step - loss: 104223.9141

Epoch 2/4

34897/34897 [==============================] - 51s 1ms/step - loss: 89332.9219

Epoch 3/4

34897/34897 [==============================] - 41s 1ms/step - loss: 88724.3672

Epoch 4/4

34897/34897 [==============================] - 41s 1ms/step - loss: 88476.4297

3878/3878 [==============================] - 3s 807us/step

Epoch 1/4

34897/34897 [==============================] - 42s 1ms/step - loss: 104903.0547

Epoch 2/4

34897/34897 [==============================] - 41s 1ms/step - loss: 89857.8984

Epoch 3/4

34897/34897 [==============================] - 41s 1ms/step - loss: 89168.1328

Epoch 4/4

34897/34897 [==============================] - 41s 1ms/step - loss: 88917.9141

3878/3878 [==============================] - 3s 788us/step

Epoch 1/4

34897/34897 [==============================] - 42s 1ms/step - loss: 104915.8281

Epoch 2/4

34897/34897 [==============================] - 41s 1ms/step - loss: 89506.5703

Epoch 3/4

34897/34897 [==============================] - 41s 1ms/step - loss: 88801.4062

Epoch 4/4

34897/34897 [==============================] - 41s 1ms/step - loss: 88495.7266

3878/3878 [==============================] - 3s 807us/step

Epoch 1/4

34897/34897 [==============================] - 41s 1ms/step - loss: 105358.7734

Epoch 2/4

34897/34897 [==============================] - 41s 1ms/step - loss: 89421.8750

Epoch 3/4

34897/34897 [==============================] - 41s 1ms/step - loss: 88726.1094

Epoch 4/4

34897/34897 [==============================] - 40s 1ms/step - loss: 88493.3594

3878/3878 [==============================] - 6s 2ms/step

Epoch 1/4

34897/34897 [==============================] - 42s 1ms/step - loss: 105393.1875

Epoch 2/4

34897/34897 [==============================] - 42s 1ms/step - loss: 89827.2891

Epoch 3/4

34897/34897 [==============================] - 42s 1ms/step - loss: 89058.5078

Epoch 4/4

34897/34897 [==============================] - 42s 1ms/step - loss: 88771.0000

3878/3878 [==============================] - 3s 831us/step

Epoch 1/4

34897/34897 [==============================] - 43s 1ms/step - loss: 103881.5391

Epoch 2/4

34897/34897 [==============================] - 43s 1ms/step - loss: 89375.8828

Epoch 3/4

34897/34897 [==============================] - 43s 1ms/step - loss: 88844.2109

Epoch 4/4

34897/34897 [==============================] - 42s 1ms/step - loss: 88631.4141

3878/3878 [==============================] - 3s 816us/step

Epoch 1/4

34897/34897 [==============================] - 43s 1ms/step - loss: 105074.3516

Epoch 2/4

34897/34897 [==============================] - 43s 1ms/step - loss: 89607.5938

Epoch 3/4

34897/34897 [==============================] - 45s 1ms/step - loss: 88952.6797

Epoch 4/4

34897/34897 [==============================] - 44s 1ms/step - loss: 88696.1250

3878/3878 [==============================] - 4s 920us/step

Epoch 1/4

34897/34897 [==============================] - 45s 1ms/step - loss: 105804.1406

Epoch 2/4

34897/34897 [==============================] - 44s 1ms/step - loss: 89668.7422

Epoch 3/4

34897/34897 [==============================] - 44s 1ms/step - loss: 88952.0469

Epoch 4/4

34897/34897 [==============================] - 44s 1ms/step - loss: 88688.6484

3878/3878 [==============================] - 3s 866us/step

Epoch 1/4

34897/34897 [==============================] - 44s 1ms/step - loss: 105719.8438

Epoch 2/4

34897/34897 [==============================] - 44s 1ms/step - loss: 89821.7891

Epoch 3/4

34897/34897 [==============================] - 43s 1ms/step - loss: 89165.9141

Epoch 4/4

34897/34897 [==============================] - 43s 1ms/step - loss: 88894.5938

3878/3878 [==============================] - 3s 833us/step

Epoch 1/4

34897/34897 [==============================] - 44s 1ms/step - loss: 104387.6250

Epoch 2/4

34897/34897 [==============================] - 44s 1ms/step - loss: 89819.4375

Epoch 3/4

34897/34897 [==============================] - 43s 1ms/step - loss: 89099.2812

Epoch 4/4

34897/34897 [==============================] - 43s 1ms/step - loss: 88785.7734

3878/3878 [==============================] - 3s 825us/step

Mean R2 score: 0.7752077972590461

Standard deviation of R2 scores: 0.001493568094697646

Mean MSE: 88660.01931084515

Standard deviation of MSEs: 860.6918448792612

R2 score: 0.7754651961151788

MSE: 88479.19242670664

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=5, batch\_size=32)

R2 score: 0.7750406641987644

MSE: 88646.48159735555

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

R2 score: 0.7669494372517881

MSE: 91834.87472672047

15510/15510 [==============================] - 27s 2ms/step - loss: 100817.9297

7755/7755 [==============================] - 8s 958us/step

R2 score: 0.7703654493498198

MSE: 90488.77609735727

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=1, batch\_size=64)

7755/7755 [==============================] - 14s 2ms/step - loss: 113476.3906

7755/7755 [==============================] - 9s 1ms/step

R2 score: 0.7664006904942725

MSE: 92051.11144865272

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2\_score, mean\_squared\_error

from keras.models import Sequential

from keras.layers import Dense

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df.drop(['trip\_duration'], axis=1), df['trip\_duration'], test\_size=0.2, random\_state=42)

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=1, batch\_size=128)

# Use the trained model to make predictions on the testing set

y\_pred = model.predict(X\_test)

# Calculate the R2 score

r2 = r2\_score(y\_test, y\_pred)

# Calculate the MSE

mse = mean\_squared\_error(y\_test, y\_pred)

# Print the R2 score and MSE

print('R2 score:', r2)

print('MSE:', mse)

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=1, batch\_size=256)

3878/3878 [==============================] - 8s 2ms/step - loss: 134913.1094

7755/7755 [==============================] - 9s 1ms/step

R2 score: 0.7614042397720535

MSE: 94019.99073708834

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=1, batch\_size=512)

1939/1939 [==============================] - 5s 2ms/step - loss: 166352.3750

7755/7755 [==============================] - 9s 1ms/step

R2 score: 0.7602035499184304

MSE: 94493.12927403429

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=5, batch\_size=512)

Epoch 1/5

1939/1939 [==============================] - 6s 3ms/step - loss: 182104.2344

Epoch 2/5

1939/1939 [==============================] - 6s 3ms/step - loss: 95098.5703

Epoch 3/5

1939/1939 [==============================] - 6s 3ms/step - loss: 93204.4766

Epoch 4/5

1939/1939 [==============================] - 6s 3ms/step - loss: 91658.1562

Epoch 5/5

1939/1939 [==============================] - 6s 3ms/step - loss: 90697.0781

7755/7755 [==============================] - 8s 1ms/step

R2 score: 0.7685079606424498

MSE: 91220.72988771099

Epoch 1/10

1939/1939 [==============================] - 6s 3ms/step - loss: 158483.7969

Epoch 2/10

1939/1939 [==============================] - 5s 3ms/step - loss: 92164.3203

Epoch 3/10

1939/1939 [==============================] - 5s 3ms/step - loss: 90201.8047

Epoch 4/10

1939/1939 [==============================] - 5s 3ms/step - loss: 89513.7656

Epoch 5/10

1939/1939 [==============================] - 5s 3ms/step - loss: 89174.3047

Epoch 6/10

1939/1939 [==============================] - 5s 3ms/step - loss: 88969.5781

Epoch 7/10

1939/1939 [==============================] - 5s 3ms/step - loss: 88897.3438

Epoch 8/10

1939/1939 [==============================] - 6s 3ms/step - loss: 88763.1328

Epoch 9/10

1939/1939 [==============================] - 5s 3ms/step - loss: 88651.6328

Epoch 10/10

1939/1939 [==============================] - 6s 3ms/step - loss: 88603.0938

7755/7755 [==============================] - 8s 1ms/step

R2 score: 0.773548650707012

MSE: 89234.41783955933

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=10, batch\_size=512)

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=20, batch\_size=1024)

Epoch 1/20

970/970 [==============================] - 4s 3ms/step - loss: 228897.4531

Epoch 2/20

970/970 [==============================] - 3s 3ms/step - loss: 96426.3359

Epoch 3/20

970/970 [==============================] - 3s 3ms/step - loss: 94003.6562

Epoch 4/20

970/970 [==============================] - 3s 3ms/step - loss: 92464.7422

Epoch 5/20

970/970 [==============================] - 3s 3ms/step - loss: 91273.0469

Epoch 6/20

970/970 [==============================] - 3s 3ms/step - loss: 90582.4375

Epoch 7/20

970/970 [==============================] - 3s 3ms/step - loss: 90085.5391

Epoch 8/20

970/970 [==============================] - 3s 4ms/step - loss: 89686.2656

Epoch 9/20

970/970 [==============================] - 3s 3ms/step - loss: 89434.3750

Epoch 10/20

970/970 [==============================] - 3s 3ms/step - loss: 89193.9531

Epoch 11/20

970/970 [==============================] - 3s 3ms/step - loss: 89020.6641

Epoch 12/20

970/970 [==============================] - 3s 3ms/step - loss: 88917.6328

Epoch 13/20

970/970 [==============================] - 3s 3ms/step - loss: 88789.4531

Epoch 14/20

970/970 [==============================] - 3s 3ms/step - loss: 88736.7188

Epoch 15/20

970/970 [==============================] - 3s 4ms/step - loss: 88668.3203

Epoch 16/20

970/970 [==============================] - 4s 4ms/step - loss: 88599.0078

Epoch 17/20

970/970 [==============================] - 3s 3ms/step - loss: 88564.7266

Epoch 18/20

970/970 [==============================] - 3s 3ms/step - loss: 88516.2266

Epoch 19/20

970/970 [==============================] - 3s 3ms/step - loss: 88489.1797

Epoch 20/20

970/970 [==============================] - 3s 3ms/step - loss: 88462.6328

7755/7755 [==============================] - 8s 1ms/step

R2 score: 0.7752196016262214

MSE: 88575.97030555407

# Define the deep learning model

model = Sequential()

model.add(Dense(64, activation='relu', input\_dim=X\_train.shape[1]))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Train the model on the training set

model.fit(X\_train, y\_train, epochs=20, batch\_size=32)

Code includes juypter files of each steps of the entire pipeline

Report includes the report in docx and pdf formatting for convenience

Utils contains common utils used within the project, including screenshots, plugins, testing files and coursework brief