data-madrid

May 2, 2024

zzc_ail kısmı bu kodlar aracılığı ile yapıldı.

[1]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/zzc_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)


veri = veri[['open', 'low', 'high', 'close']]

X = veri[['open', 'high', 'close']].values
y = veri['open'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
 ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=50, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)

tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/50
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
360/360              1s 819us/step -
loss: 0.0122
Epoch 2/50
360/360              0s 743us/step -
loss: 1.7933e-06
Epoch 3/50
360/360              0s 820us/step -
loss: 7.6221e-07
Epoch 4/50
360/360              1s 2ms/step -
loss: 4.4084e-07
Epoch 5/50
360/360              1s 2ms/step -
loss: 4.5971e-07
Epoch 6/50
360/360              1s 2ms/step -
loss: 2.7161e-07
Epoch 7/50
```

```
360/360              1s 1ms/step -
loss: 4.7412e-07
Epoch 8/50
360/360              1s 1ms/step -
loss: 2.4705e-07
Epoch 9/50
360/360              1s 2ms/step -
loss: 4.2063e-07
Epoch 10/50
360/360              1s 2ms/step -
loss: 3.2261e-07
Epoch 11/50
360/360              1s 2ms/step -
loss: 2.9180e-07
Epoch 12/50
360/360              1s 2ms/step -
loss: 5.1377e-07
Epoch 13/50
360/360              1s 2ms/step -
loss: 4.9525e-07
Epoch 14/50
360/360              1s 2ms/step -
loss: 2.5193e-06
Epoch 15/50
360/360              1s 2ms/step -
loss: 7.7375e-07
Epoch 16/50
360/360              1s 2ms/step -
loss: 2.8358e-07
Epoch 17/50
360/360              1s 2ms/step -
loss: 2.9692e-07
Epoch 18/50
360/360              1s 2ms/step -
loss: 8.0565e-07
Epoch 19/50
360/360              1s 2ms/step -
loss: 2.9873e-06
Epoch 20/50
360/360              1s 2ms/step -
loss: 1.8617e-07
Epoch 21/50
360/360              1s 2ms/step -
loss: 2.6134e-06
Epoch 22/50
360/360              1s 2ms/step -
loss: 3.4818e-07
Epoch 23/50
```

```
360/360              1s 2ms/step -
loss: 2.2442e-06
Epoch 24/50
360/360              1s 2ms/step -
loss: 1.1519e-06
Epoch 25/50
360/360              1s 2ms/step -
loss: 4.1131e-07
Epoch 26/50
360/360              1s 2ms/step -
loss: 8.1102e-07
Epoch 27/50
360/360              1s 2ms/step -
loss: 2.1929e-06
Epoch 28/50
360/360              1s 2ms/step -
loss: 5.6680e-07
Epoch 29/50
360/360              1s 2ms/step -
loss: 2.8018e-06
Epoch 30/50
360/360              1s 2ms/step -
loss: 8.3587e-07
Epoch 31/50
360/360              1s 2ms/step -
loss: 2.1951e-07
Epoch 32/50
360/360              1s 1ms/step -
loss: 3.1545e-06
Epoch 33/50
360/360              1s 2ms/step -
loss: 1.5138e-06
Epoch 34/50
360/360              1s 2ms/step -
loss: 3.4526e-06
Epoch 35/50
360/360              1s 2ms/step -
loss: 8.7778e-08
Epoch 36/50
360/360              1s 2ms/step -
loss: 2.1773e-07
Epoch 37/50
360/360              1s 2ms/step -
loss: 2.2318e-06
Epoch 38/50
360/360              1s 2ms/step -
loss: 2.7593e-07
Epoch 39/50
```

```
360/360             1s 2ms/step -
loss: 1.0940e-06
Epoch 40/50
360/360             1s 2ms/step -
loss: 5.7537e-07
Epoch 41/50
360/360             1s 2ms/step -
loss: 1.4486e-06
Epoch 42/50
360/360             1s 2ms/step -
loss: 1.0347e-07
Epoch 43/50
360/360             1s 2ms/step -
loss: 4.5939e-07
Epoch 44/50
360/360             1s 2ms/step -
loss: 3.2728e-06
Epoch 45/50
360/360             1s 2ms/step -
loss: 2.0950e-07
Epoch 46/50
360/360             1s 2ms/step -
loss: 2.0090e-07
Epoch 47/50
360/360             1s 2ms/step -
loss: 7.6189e-08
Epoch 48/50
360/360             1s 2ms/step -
loss: 2.1330e-07
Epoch 49/50
360/360             1s 2ms/step -
loss: 1.7747e-06
Epoch 50/50
360/360             1s 2ms/step -
loss: 5.7129e-07
7/7             0s 11ms/step
Gelecekteki Fiyat Tahminleri:
[[23871.344]
 [23753.27 ]
 [23651.28 ]
 [23809.049]
 [24991.963]
 [24733.021]
 [25079.967]
 [25172.107]
 [24871.201]
 [25082.748]
 [25315.697]
```

```
[24837.883]
[24953.238]
[24795.195]
[25015.973]
[25256.98 ]
[25344.06 ]
[25273.875]
[25130.021]
[25210.855]
[24994.365]
[24805.578]
[24851.637]
[24910.877]
[24994.154]
[24876.48 ]
[24778.95 ]
[24772.316]
[24765.885]
[24780.455]
[24981.574]
[25088.145]
[25113.566]
[25178.053]
[25179.555]
[25111.068]
[25181.521]
[25269.988]
[25319.484]
[25346.95 ]
[25448.535]
[25455.105]
[25419.377]
[25483.74 ]
[25296.252]
[25149.588]
[25217.81 ]
[25265.855]
[25287.059]
[25258.988]
[25180.844]
[25448.63 ]
[25341.527]
[25289.861]
[25263.48 ]
[25356.574]
[25507.584]
[25406.2  ]
[25531.955]
```

```
[25786.533]
[25455.904]
[25550.541]
[25732.371]
[25820.217]
[25955.623]
[26046.025]
[25948.787]
[25701.227]
[25928.812]
[25626.074]
[25701.766]
[25703.623]
[25788.688]
[25792.299]
[26085.506]
[26746.512]
[26781.87 ]
[26995.96 ]
[26897.016]
[26909.578]
[27074.371]
[26918.666]
[27158.908]
[27095.088]
[27291.162]
[27277.44 ]
[27276.32 ]
[27088.95 ]
[27096.646]
[27274.752]
[27222.357]
[27109.379]
[27080.688]
[27150.447]
[26930.646]
[27004.494]
[27179.621]
[26969.404]
[27055.475]
[27022.166]
[27035.557]
[27094.559]
[27126.682]
[26936.21 ]
[26818.732]
[26398.543]
[26557.182]
```

```
[26682.508]
[26697.25 ]
[26567.094]
[26506.072]
[26643.422]
[26786.342]
[26857.982]
[27110.803]
[27050.838]
[27100.887]
[27163.854]
[27147.258]
[27356.756]
[27415.822]
[27635.863]
[27582.988]
[27555.629]
[27493.754]
[27401.088]
[27156.578]
[27231.318]
[27343.23 ]
[27479.996]
[27532.678]
[27622.541]
[27463.475]
[27576.043]
[27448.834]
[27516.258]
[27406.219]
[27442.238]
[27480.22 ]
[27567.787]
[27475.6  ]
[27525.951]
[27689.45 ]
[27558.957]
[27605.277]
[27461.03 ]
[27431.576]
[27420.547]
[27033.71 ]
[26952.512]
[26626.805]
[26800.555]
[26704.191]
[26351.137]
[26058.736]
```

```
 [26559.285]
 [26720.146]
 [26894.479]
 [26839.03 ]
 [27030.   ]
 [26920.787]
 [25949.826]
 [25640.41 ]
 [25342.227]
 [25803.71 ]
 [25716.193]
 [25203.31 ]
 [25473.576]
 [25278.988]
 [25456.709]
 [25514.707]
 [25771.562]
 [25797.584]
 [25621.287]
 [25529.914]
 [25649.312]
 [26395.55 ]
 [25934.72 ]
 [25607.486]
 [25655.39 ]
 [25669.967]
 [25722.031]
 [25288.902]
 [25472.633]
 [25542.414]
 [25826.389]
 [25746.607]
 [25637.082]
 [25567.078]
 [25500.55 ]
 [25597.553]
 [25764.291]
 [25396.95 ]
 [25179.342]
 [25373.682]
 [25241.191]
 [25340.871]
 [25195.248]
 [25180.865]
 [25420.088]
 [25459.646]]
72/72             0s 2ms/step
RMSE: 0.00048756292988456283
```

```
MAE: 0.00035567596396486546
```

```
[2]: import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/zzc_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)



veri = veri[['open', 'low', 'high', 'close']]

X = veri[['open', 'high', 'low']].values
y = veri['high'].values


scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)


X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)


model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
 ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])


model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_scaled, y_scaled, epochs=25, batch_size=32, verbose=1)


gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)


tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)


y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/25

C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

360/360            3s 2ms/step -
loss: 0.0126
Epoch 2/25
360/360            1s 2ms/step -
loss: 3.0413e-06
Epoch 3/25
360/360            1s 2ms/step -
loss: 2.1935e-06
Epoch 4/25
360/360            1s 2ms/step -
loss: 1.6653e-06
Epoch 5/25
360/360            1s 2ms/step -
loss: 3.1726e-06
Epoch 6/25
360/360            1s 2ms/step -
loss: 1.7359e-06
Epoch 7/25
```

```
360/360             1s 2ms/step -
loss: 2.0367e-06
Epoch 8/25
360/360             1s 2ms/step -
loss: 1.5843e-06
Epoch 9/25
360/360             1s 2ms/step -
loss: 1.7682e-06
Epoch 10/25
360/360             1s 2ms/step -
loss: 1.0920e-06
Epoch 11/25
360/360             1s 2ms/step -
loss: 1.2651e-06
Epoch 12/25
360/360             1s 2ms/step -
loss: 1.3778e-06
Epoch 13/25
360/360             1s 2ms/step -
loss: 1.3147e-06
Epoch 14/25
360/360             1s 2ms/step -
loss: 6.6282e-07
Epoch 15/25
360/360             1s 2ms/step -
loss: 1.8652e-06
Epoch 16/25
360/360             1s 2ms/step -
loss: 9.7717e-07
Epoch 17/25
360/360             1s 2ms/step -
loss: 1.9848e-06
Epoch 18/25
360/360             1s 2ms/step -
loss: 6.4578e-07
Epoch 19/25
360/360             1s 2ms/step -
loss: 6.3957e-07
Epoch 20/25
360/360             1s 2ms/step -
loss: 5.3102e-06
Epoch 21/25
360/360             1s 2ms/step -
loss: 7.9311e-07
Epoch 22/25
360/360             1s 2ms/step -
loss: 1.0619e-06
Epoch 23/25
```

```
360/360             1s 2ms/step -
loss: 8.0312e-07
Epoch 24/25
360/360             1s 2ms/step -
loss: 1.0719e-06
Epoch 25/25
360/360             1s 2ms/step -
loss: 1.7251e-07
7/7            0s 12ms/step
Gelecekteki Fiyat Tahminleri:
[[24046.928]
 [23932.057]
 [23889.87 ]
 [24349.926]
 [25069.434]
 [25149.174]
 [25482.625]
 [25373.389]
 [25160.959]
 [25360.232]
 [25418.852]
 [25101.629]
 [25337.133]
 [25122.69 ]
 [25308.984]
 [25388.154]
 [25438.852]
 [25361.744]
 [25426.988]
 [25249.803]
 [25059.016]
 [24967.352]
 [24968.21 ]
 [25018.826]
 [25049.662]
 [25016.379]
 [24931.635]
 [24924.873]
 [24947.703]
 [25048.5  ]
 [25261.576]
 [25212.977]
 [25237.479]
 [25350.605]
 [25298.229]
 [25263.283]
 [25609.81 ]
 [25480.148]
```

```
[25453.895]
[25648.027]
[25575.006]
[25596.135]
[25643.037]
[25502.871]
[25453.936]
[25318.05 ]
[25311.266]
[25361.947]
[25358.121]
[25308.246]
[25494.352]
[25514.73 ]
[25434.848]
[25436.016]
[25429.379]
[11132.418]
[25557.479]
[11130.018]
[25899.273]
[26421.729]
[25608.994]
[25961.127]
[25962.99 ]
[25991.516]
[26191.492]
[26097.484]
[25979.248]
[25980.521]
[25974.105]
[25772.209]
[11133.313]
[25864.662]
[25928.475]
[26129.412]
[26854.605]
[27003.566]
[27066.006]
[27063.89 ]
[27229.164]
[27205.473]
[27272.914]
[27275.754]
[27403.023]
[27368.988]
[27355.873]
[27375.992]
```

```
[27443.08 ]
[27241.29 ]
[27356.72 ]
[27422.012]
[27296.762]
[27266.268]
[27307.887]
[27211.633]
[27140.678]
[27153.809]
[27262.057]
[27169.418]
[27129.303]
[27222.326]
[27140.877]
[27175.143]
[27382.564]
[27040.72 ]
[26890.078]
[26538.996]
[11128.871]
[26836.416]
[26738.725]
[26665.18 ]
[26722.549]
[26841.928]
[27033.307]
[27166.58 ]
[27160.111]
[27216.842]
[27212.84 ]
[27211.838]
[27412.395]
[27589.861]
[27765.088]
[27745.398]
[27664.535]
[27688.316]
[27582.492]
[27515.088]
[27485.168]
[27559.348]
[27532.494]
[27597.326]
[27679.152]
[27694.021]
[27674.209]
[27624.672]
```

```
[27573.824]
[27584.559]
[27499.799]
[27600.336]
[27635.666]
[27760.541]
[27583.164]
[27749.43 ]
[27814.35 ]
[27724.78 ]
[27710.96 ]
[27523.668]
[27530.555]
[27463.398]
[27310.78 ]
[27130.41 ]
[26876.334]
[11126.783]
[26723.309]
[26503.502]
[26777.01 ]
[26768.312]
[26983.33 ]
[27006.377]
[27277.068]
[27134.006]
[26956.426]
[26122.82 ]
[25763.47 ]
[25686.469]
[25913.932]
[25795.492]
[25598.705]
[25619.457]
[25539.871]
[11129.792]
[25851.457]
[25842.572]
[11126.219]
[25762.377]
[11129.691]
[26517.143]
[26558.844]
[25954.229]
[25777.502]
[25951.066]
[25869.885]
[25746.61 ]
```

```
 [25515.67 ]
 [25658.748]
 [25801.068]
 [25868.94 ]
 [25785.473]
 [25666.576]
 [25687.86 ]
 [25666.125]
 [25782.975]
 [25811.164]
 [25514.508]
 [25413.113]
 [25627.424]
 [25407.688]
 [25402.021]
 [25400.379]
 [25380.805]
 [25506.182]
 [25610.861]]
72/72             0s 2ms/step
RMSE: 0.000330938398256279
MAE: 0.0002776021253427656
```

[3]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/zzc_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)


veri = veri[['open', 'low', 'high', 'close']]

X = veri[['open', 'high', 'low']].values
y = veri['low'].values
```

```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
 ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=25, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)


tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)


y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/25
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
360/360              3s 2ms/step -
loss: 0.0185
Epoch 2/25
360/360              1s 2ms/step -
```

```
loss: 3.2344e-06
Epoch 3/25
360/360              1s 2ms/step -
loss: 2.7664e-06
Epoch 4/25
360/360              1s 2ms/step -
loss: 2.3211e-06
Epoch 5/25
360/360              1s 2ms/step -
loss: 2.0229e-06
Epoch 6/25
360/360              1s 2ms/step -
loss: 1.4939e-06
Epoch 7/25
360/360              1s 2ms/step -
loss: 2.2801e-06
Epoch 8/25
360/360              1s 2ms/step -
loss: 1.3525e-06
Epoch 9/25
360/360              1s 2ms/step -
loss: 1.0585e-06
Epoch 10/25
360/360              1s 2ms/step -
loss: 1.0616e-06
Epoch 11/25
360/360              1s 2ms/step -
loss: 1.2644e-06
Epoch 12/25
360/360              1s 2ms/step -
loss: 1.0310e-06
Epoch 13/25
360/360              1s 2ms/step -
loss: 7.1882e-07
Epoch 14/25
360/360              1s 2ms/step -
loss: 4.0542e-07
Epoch 15/25
360/360              1s 2ms/step -
loss: 5.7388e-07
Epoch 16/25
360/360              1s 2ms/step -
loss: 1.8575e-06
Epoch 17/25
360/360              1s 2ms/step -
loss: 3.8119e-06
Epoch 18/25
360/360              1s 2ms/step -
```

```
loss: 5.0710e-07
Epoch 19/25
360/360             1s 2ms/step -
loss: 4.2924e-07
Epoch 20/25
360/360             1s 2ms/step -
loss: 6.8676e-07
Epoch 21/25
360/360             1s 2ms/step -
loss: 1.7266e-06
Epoch 22/25
360/360             1s 2ms/step -
loss: 2.9850e-07
Epoch 23/25
360/360             1s 2ms/step -
loss: 1.0493e-06
Epoch 24/25
360/360             1s 2ms/step -
loss: 1.3268e-06
Epoch 25/25
360/360             1s 2ms/step -
loss: 1.9646e-06
7/7            0s 11ms/step
Gelecekteki Fiyat Tahminleri:
[[23852.3  ]
 [23744.932]
 [23738.182]
 [23896.307]
 [24853.064]
 [24757.57 ]
 [25148.582]
 [24977.754]
 [24979.156]
 [25093.547]
 [24524.24 ]
 [24704.115]
 [24904.135]
 [24678.873]
 [25042.2  ]
 [25219.197]
 [25221.748]
 [25215.26 ]
 [25220.307]
 [25080.385]
 [24923.906]
 [24840.404]
 [24837.154]
 [24791.48 ]
```

```
[24913.256]
[24885.273]
[24814.521]
[24829.98 ]
[24827.701]
[24863.86 ]
[25076.365]
[25153.531]
[25195.695]
[25140.914]
[25180.78 ]
[25173.85 ]
[25170.715]
[25234.309]
[25355.041]
[25432.236]
[25506.81 ]
[25441.064]
[25517.6  ]
[25366.98 ]
[25309.502]
[25223.88 ]
[25276.1  ]
[25308.258]
[25336.234]
[25263.42 ]
[25255.16 ]
[25425.701]
[25361.271]
[25356.592]
[25370.578]
[25386.643]
[25472.42 ]
[25385.94 ]
[25568.303]
[25477.12 ]
[25021.66 ]
[25613.8  ]
[25770.752]
[25800.895]
[26039.697]
[26076.186]
[25750.828]
[25805.418]
[25390.418]
[25568.922]
[25748.037]
[25798.354]
```

```
[25880.229]
[25859.67 ]
[26162.637]
[26824.83 ]
[26889.35 ]
[27008.766]
[26954.225]
[26980.77 ]
[26966.834]
[27033.477]
[27185.467]
[27136.303]
[27270.166]
[27278.307]
[27155.6  ]
[27165.457]
[27163.21 ]
[27356.703]
[27219.496]
[27188.908]
[27160.67 ]
[26998.777]
[26936.592]
[27078.424]
[27203.605]
[27081.412]
[27062.139]
[27122.324]
[26998.562]
[27069.516]
[27044.459]
[26840.61 ]
[26466.205]
[25986.719]
[26473.17 ]
[26743.309]
[26633.576]
[26600.562]
[26598.799]
[26726.592]
[26909.178]
[26931.576]
[27140.646]
[27151.105]
[27173.822]
[27217.637]
[27245.576]
[27449.127]
```

```
[27512.898]
[27650.084]
[27630.16 ]
[27600.916]
[27401.508]
[27111.668]
[27226.494]
[27320.1  ]
[27337.36 ]
[27481.152]
[27537.234]
[27571.566]
[27563.652]
[27522.127]
[27543.326]
[27515.865]
[27416.678]
[27530.135]
[27603.197]
[27497.723]
[27514.297]
[27638.768]
[27649.46 ]
[27678.531]
[27666.834]
[27469.65 ]
[27385.79 ]
[27029.871]
[26910.807]
[26672.645]
[26399.113]
[26635.459]
[26220.137]
[25919.23 ]
[26121.115]
[26529.426]
[26839.955]
[26967.764]
[26900.709]
[27046.533]
[25736.566]
[25252.803]
[25282.57 ]
[25343.936]
[25739.049]
[25326.615]
[24790.73 ]
[25394.   ]
```

```
 [25204.459]
 [25432.205]
 [25464.102]
 [25639.758]
 [25697.875]
 [25512.102]
 [25501.35 ]
 [25727.537]
 [25926.783]
 [25514.77 ]
 [25723.678]
 [25768.541]
 [25742.22 ]
 [25423.459]
 [25228.459]
 [25581.133]
 [25558.52 ]
 [25837.69 ]
 [25732.902]
 [25433.098]
 [25550.713]
 [25566.084]
 [25673.744]
 [25397.979]
 [25186.219]
 [25166.088]
 [25304.727]
 [25153.596]
 [25299.186]
 [25263.076]
 [25293.35 ]
 [25420.727]
 [25513.229]]
72/72                 0s 1ms/step
RMSE: 0.0011752951749601454
MAE: 0.0008784334568683738
```

[4]:
```python
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/zzc_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
```

```
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)


veri = veri[['open', 'low', 'high', 'close']]

X = veri[['open', 'high', 'low']].values
y = veri['close'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
↪test_size=0.2, random_state=42)


model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])


model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=25, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]  # Tüm verilerin son 89 verisini kullanıyoruz
tahminler = model.predict(gelecek_veri)

tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
```

```python
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/25

C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

360/360             3s 2ms/step -
loss: 0.0055
Epoch 2/25
360/360             1s 2ms/step -
loss: 0.0021
Epoch 3/25
360/360             1s 2ms/step -
loss: 0.0018
Epoch 4/25
360/360             1s 2ms/step -
loss: 0.0020
Epoch 5/25
360/360             1s 2ms/step -
loss: 0.0021
Epoch 6/25
360/360             1s 2ms/step -
loss: 0.0020
Epoch 7/25
360/360             1s 2ms/step -
loss: 0.0021
Epoch 8/25
360/360             1s 2ms/step -
loss: 0.0019
Epoch 9/25
360/360             1s 2ms/step -
loss: 0.0022
Epoch 10/25
360/360             1s 2ms/step -
loss: 0.0018
Epoch 11/25
360/360             1s 2ms/step -
loss: 0.0017
Epoch 12/25
360/360             1s 1ms/step -
loss: 0.0020
Epoch 13/25
360/360             1s 2ms/step -
```

```
loss: 0.0022
Epoch 14/25
360/360             1s 1ms/step -
loss: 0.0019
Epoch 15/25
360/360             1s 1ms/step -
loss: 0.0017
Epoch 16/25
360/360             1s 1ms/step -
loss: 0.0021
Epoch 17/25
360/360             1s 1ms/step -
loss: 0.0021
Epoch 18/25
360/360             1s 1ms/step -
loss: 0.0020
Epoch 19/25
360/360             1s 1ms/step -
loss: 0.0023
Epoch 20/25
360/360             1s 1ms/step -
loss: 0.0023
Epoch 21/25
360/360             1s 1ms/step -
loss: 0.0018
Epoch 22/25
360/360             1s 1ms/step -
loss: 0.0023
Epoch 23/25
360/360             1s 1ms/step -
loss: 0.0019
Epoch 24/25
360/360             1s 2ms/step -
loss: 0.0018
Epoch 25/25
360/360             1s 2ms/step -
loss: 0.0020
7/7             0s 12ms/step
Gelecekteki Fiyat Tahminleri:
[[22398.5  ]
 [22296.781]
 [22259.416]
 [22454.293]
 [23343.756]
 [23243.453]
 [23577.275]
 [23502.238]
 [23387.04 ]
```

```
[23531.291]
[23329.557]
[23227.174]
[23403.05 ]
[23218.203]
[23480.533]
[23641.809]
[23674.71 ]
[23639.84 ]
[23615.111]
[23540.223]
[23376.188]
[23270.998]
[23281.873]
[23284.605]
[23369.55 ]
[23319.342]
[23245.57 ]
[23249.979]
[23251.07 ]
[23289.248]
[23479.434]
[23536.045]
[23566.945]
[23577.33 ]
[23587.682]
[23560.318]
[23636.326]
[23658.148]
[23733.852]
[23810.648]
[23860.564]
[23834.748]
[23869.453]
[23784.695]
[23706.098]
[23603.559]
[23645.254]
[23681.838]
[23700.053]
[23649.645]
[23656.654]
[23811.914]
[23739.418]
[23723.646]
[23722.172]
[24595.209]
[23856.977]
```

```
[24619.748]
[23966.97 ]
[24068.727]
[23638.729]
[24003.91 ]
[24126.848]
[24169.371]
[24352.326]
[24377.621]
[24177.627]
[24138.02 ]
[24000.844]
[23970.684]
[24984.43 ]
[24115.607]
[24187.77 ]
[24213.21 ]
[24561.465]
[25061.975]
[25123.455]
[25235.727]
[25213.75 ]
[25225.018]
[25272.158]
[25265.979]
[25421.371]
[25375.94 ]
[25487.58 ]
[25491.195]
[25444.1  ]
[25364.791]
[25386.824]
[25535.37 ]
[25435.213]
[25385.527]
[25372.328]
[25297.115]
[25197.287]
[25286.295]
[25410.457]
[25281.01 ]
[25287.955]
[25323.363]
[25254.576]
[25309.654]
[25342.129]
[25127.793]
[24902.135]
```

```
[24504.094]
[25857.479]
[24988.135]
[24923.488]
[24860.79 ]
[24853.533]
[24970.82 ]
[25127.691]
[25181.5  ]
[25338.672]
[25343.715]
[25367.117]
[25404.404]
[25449.229]
[25631.543]
[25710.16 ]
[25825.346]
[25787.97 ]
[25771.336]
[25642.053]
[25468.703]
[25456.299]
[25533.506]
[25563.82 ]
[25678.611]
[25733.629]
[25775.719]
[25728.486]
[25728.182]
[25695.889]
[25699.512]
[25611.975]
[25693.064]
[25744.053]
[25738.83 ]
[25686.578]
[25794.088]
[25851.297]
[25816.326]
[25816.658]
[25655.652]
[25609.29 ]
[25422.734]
[25241.873]
[25076.246]
[24817.256]
[26087.295]
[24726.951]
```

```
[24453.3  ]
[24523.06 ]
[24842.635]
[25068.916]
[25179.574]
[25183.36 ]
[25274.545]
[24582.74 ]
[23964.86 ]
[23837.309]
[23774.064]
[24122.44 ]
[23883.639]
[23456.344]
[23821.264]
[23666.46 ]
[24673.162]
[23904.941]
[24047.078]
[25003.928]
[23940.768]
[24750.49 ]
[24173.76 ]
[24478.002]
[24057.836]
[24039.863]
[24103.256]
[24080.95 ]
[23922.79 ]
[23676.326]
[23916.38 ]
[23948.422]
[24167.537]
[24082.541]
[23891.357]
[23931.99 ]
[23917.955]
[24014.52 ]
[23932.953]
[23684.727]
[23600.287]
[23753.803]
[23609.873]
[23704.23 ]
[23648.152]
[23655.424]
[23800.592]
[23872.613]]
```

```
72/72              0s 1ms/step
RMSE: 0.05041986269688317
MAE: 0.023680437711474354
```

ail_frx kısmı bu kodlar aracılığı ile yapıldı.

```
[5]: import pandas as pd
     import numpy as np
     import tensorflow as tf
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.metrics import mean_squared_error, mean_absolute_error

     veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/ail_frx.csv")

     veri['open'] = veri['open'].str.replace('"', '').astype(float)
     veri['high'] = veri['high'].str.replace('"', '').astype(float)
     veri['low'] = veri['low'].str.replace('"', '').astype(float)
     veri['close'] = veri['close'].str.replace('"', '').astype(float)

     veri['open'].fillna(veri['open'].mean(), inplace=True)
     veri['high'].fillna(veri['high'].mean(), inplace=True)
     veri['low'].fillna(veri['low'].mean(), inplace=True)
     veri['close'].fillna(veri['close'].mean(), inplace=True)


     veri = veri[['open', 'low', 'high', 'close']]

     X = veri[['close', 'open', 'high']].values
     y = veri['open'].values

     scaler = MinMaxScaler()
     X_scaled = scaler.fit_transform(X)
     y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

     X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
      ↪test_size=0.2, random_state=42)

     model = tf.keras.models.Sequential([
         tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
      ↪shape[1],)),
         tf.keras.layers.Dense(32, activation='relu'),
         tf.keras.layers.Dense(1)
     ])

     model.compile(optimizer='adam', loss='mean_squared_error')

     model.fit(X_scaled, y_scaled, epochs=25, batch_size=32, verbose=1)
```

```
gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)

tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/25
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
354/354              3s 2ms/step -
loss: 0.0314
Epoch 2/25
354/354              1s 2ms/step -
loss: 2.5997e-06
Epoch 3/25
354/354              1s 2ms/step -
loss: 2.7128e-06
Epoch 4/25
354/354              1s 2ms/step -
loss: 1.9581e-06
Epoch 5/25
354/354              1s 2ms/step -
loss: 2.0711e-06
Epoch 6/25
354/354              1s 2ms/step -
loss: 2.1813e-06
Epoch 7/25
354/354              1s 2ms/step -
loss: 2.1230e-06
Epoch 8/25
354/354              1s 2ms/step -
loss: 1.9547e-06
Epoch 9/25
```

```
354/354              1s 2ms/step -
loss: 2.2674e-06
Epoch 10/25
354/354              1s 2ms/step -
loss: 3.5260e-06
Epoch 11/25
354/354              1s 2ms/step -
loss: 2.4798e-06
Epoch 12/25
354/354              1s 2ms/step -
loss: 1.6496e-06
Epoch 13/25
354/354              1s 2ms/step -
loss: 2.5276e-06
Epoch 14/25
354/354              1s 2ms/step -
loss: 2.1326e-06
Epoch 15/25
354/354              1s 2ms/step -
loss: 2.3528e-06
Epoch 16/25
354/354              1s 2ms/step -
loss: 2.2273e-06
Epoch 17/25
354/354              1s 2ms/step -
loss: 1.7572e-06
Epoch 18/25
354/354              1s 2ms/step -
loss: 2.0166e-06
Epoch 19/25
354/354              1s 2ms/step -
loss: 2.8404e-06
Epoch 20/25
354/354              1s 2ms/step -
loss: 3.9821e-06
Epoch 21/25
354/354              1s 2ms/step -
loss: 2.4832e-06
Epoch 22/25
354/354              1s 1ms/step -
loss: 1.6092e-06
Epoch 23/25
354/354              1s 2ms/step -
loss: 1.9422e-06
Epoch 24/25
354/354              1s 2ms/step -
loss: 1.7988e-06
Epoch 25/25
```

```
354/354             1s 2ms/step -
loss: 5.2469e-06
7/7             0s 11ms/step
Gelecekteki Fiyat Tahminleri:
[[14.265686 ]
 [14.264134 ]
 [14.263401 ]
 [14.283654 ]
 [14.2631035]
 [14.2626915]
 [14.261282 ]
 [14.262968 ]
 [14.309646 ]
 [14.266759 ]
 [14.267619 ]
 [14.269239 ]
 [14.272859 ]
 [14.273415 ]
 [14.278101 ]
 [14.280927 ]
 [14.282512 ]
 [14.281318 ]
 [14.282728 ]
 [14.286574 ]
 [14.287751 ]
 [14.280328 ]
 [14.296904 ]
 [14.300331 ]
 [14.299552 ]
 [14.299396 ]
 [14.305658 ]
 [14.311224 ]
 [14.314793 ]
 [14.3160515]
 [14.318283 ]
 [14.318148 ]
 [14.320589 ]
 [14.320487 ]
 [14.31974  ]
 [14.318864 ]
 [14.317104 ]
 [14.319963 ]
 [14.31289  ]
 [14.311619 ]
 [14.316535 ]
 [14.312172 ]
 [14.318509 ]
 [14.339562 ]
```

```
[14.361766 ]
[14.362633 ]
[14.363131 ]
[14.365618 ]
[14.36849  ]
[14.367969 ]
[14.364935 ]
[14.368728 ]
[14.368736 ]
[14.368006 ]
[14.377705 ]
[14.371225 ]
[14.373147 ]
[14.373659 ]
[14.375245 ]
[14.377011 ]
[14.361865 ]
[14.359163 ]
[14.351724 ]
[14.357038 ]
[14.380824 ]
[14.391841 ]
[14.393525 ]
[14.393555 ]
[14.391149 ]
[14.390387 ]
[14.389868 ]
[14.389391 ]
[14.39129  ]
[14.389207 ]
[14.395382 ]
[14.390063 ]
[14.389803 ]
[14.38999  ]
[14.389649 ]
[14.391044 ]
[14.416995 ]
[14.39312  ]
[14.394411 ]
[14.396838 ]
[14.392902 ]
[14.393635 ]
[14.396023 ]
[14.404543 ]
[14.4111395]
[14.41001  ]
[14.408494 ]
[14.4072485]
```

```
[14.40841  ]
[14.409466 ]
[14.409305 ]
[14.4084015]
[14.408863 ]
[14.409138 ]
[14.409238 ]
[14.409391 ]
[14.409852 ]
[14.4096365]
[14.411124 ]
[14.413535 ]
[14.414159 ]
[14.418303 ]
[14.415914 ]
[14.483509 ]
[14.418657 ]
[14.409276 ]
[14.416183 ]
[14.445926 ]
[14.435576 ]
[14.458225 ]
[14.495775 ]
[14.45127  ]
[14.496017 ]
[14.446349 ]
[14.429961 ]
[14.431455 ]
[14.430305 ]
[14.431639 ]
[14.431279 ]
[14.431582 ]
[14.431098 ]
[14.432488 ]
[14.433274 ]
[14.43289  ]
[14.432374 ]
[14.432464 ]
[14.429837 ]
[14.42838  ]
[14.427289 ]
[14.42974  ]
[14.438323 ]
[14.453854 ]
[14.445852 ]
[14.446012 ]
[14.4451275]
[14.4458275]
```

```
[14.445273 ]
[14.444143 ]
[14.445065 ]
[14.446023 ]
[14.445467 ]
[14.446286 ]
[14.4453745]
[14.444784 ]
[14.445197 ]
[14.442513 ]
[14.444031 ]
[14.446415 ]
[14.446733 ]
[14.45224  ]
[14.461165 ]
[14.472392 ]
[14.480552 ]
[14.479935 ]
[14.54758  ]
[14.57854  ]
[14.521246 ]
[14.490207 ]
[14.536542 ]
[14.484169 ]
[14.48731  ]
[14.484205 ]
[14.484649 ]
[14.48595  ]
[14.48663  ]
[14.48737  ]
[14.487092 ]
[14.485684 ]
[14.485431 ]
[14.483172 ]
[14.472586 ]
[14.466604 ]
[14.456683 ]
[14.477485 ]
[14.506206 ]
[14.527969 ]
[14.526661 ]
[14.527497 ]
[14.52143  ]
[14.542024 ]
[14.523696 ]
[14.523769 ]
[14.523716 ]
[14.523457 ]
```

```
 [14.524108 ]
 [14.524186 ]
 [14.524357 ]
 [14.525089 ]
 [14.524102 ]
 [14.523304 ]
 [14.519765 ]
 [14.524507 ]
 [14.523908 ]
 [14.535981 ]
 [14.535927 ]
 [14.540493 ]
 [14.541479 ]]
71/71              0s 1ms/step
RMSE: 0.0011945772846795094
MAE: 0.0007662062744950194
```

[6]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/ail_frx.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)


veri = veri[['open', 'low', 'high', 'close']]

X = veri[['open', 'low', 'high']].values
y = veri['high'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])


model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=40, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)


tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)


y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/40

C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

354/354             3s 2ms/step -
loss: 0.0096
Epoch 2/40
354/354             1s 2ms/step -
loss: 1.3635e-06
Epoch 3/40
354/354             1s 2ms/step -
loss: 1.4124e-06
```

```
Epoch 4/40
354/354             1s 2ms/step -
loss: 1.0876e-06
Epoch 5/40
354/354             1s 2ms/step -
loss: 1.3421e-06
Epoch 6/40
354/354             1s 2ms/step -
loss: 1.2658e-06
Epoch 7/40
354/354             1s 2ms/step -
loss: 8.4336e-07
Epoch 8/40
354/354             1s 2ms/step -
loss: 8.8987e-07
Epoch 9/40
354/354             1s 2ms/step -
loss: 6.6911e-07
Epoch 10/40
354/354             1s 2ms/step -
loss: 5.3682e-07
Epoch 11/40
354/354             1s 2ms/step -
loss: 6.5779e-07
Epoch 12/40
354/354             1s 2ms/step -
loss: 7.0393e-07
Epoch 13/40
354/354             1s 2ms/step -
loss: 6.4482e-07
Epoch 14/40
354/354             1s 2ms/step -
loss: 4.2942e-07
Epoch 15/40
354/354             1s 2ms/step -
loss: 5.0371e-07
Epoch 16/40
354/354             1s 2ms/step -
loss: 9.2690e-07
Epoch 17/40
354/354             1s 2ms/step -
loss: 1.0376e-06
Epoch 18/40
354/354             1s 2ms/step -
loss: 1.8039e-06
Epoch 19/40
354/354             1s 2ms/step -
loss: 1.1519e-06
```

```
Epoch 20/40
354/354             1s 2ms/step -
loss: 3.5819e-06
Epoch 21/40
354/354             1s 2ms/step -
loss: 1.7386e-06
Epoch 22/40
354/354             1s 2ms/step -
loss: 1.0832e-06
Epoch 23/40
354/354             1s 2ms/step -
loss: 4.2494e-06
Epoch 24/40
354/354             1s 2ms/step -
loss: 5.6845e-07
Epoch 25/40
354/354             1s 2ms/step -
loss: 1.4131e-06
Epoch 26/40
354/354             1s 2ms/step -
loss: 1.4671e-06
Epoch 27/40
354/354             1s 2ms/step -
loss: 1.2906e-06
Epoch 28/40
354/354             1s 2ms/step -
loss: 9.4586e-07
Epoch 29/40
354/354             1s 2ms/step -
loss: 4.0270e-07
Epoch 30/40
354/354             1s 2ms/step -
loss: 2.2781e-06
Epoch 31/40
354/354             1s 2ms/step -
loss: 4.4542e-07
Epoch 32/40
354/354             1s 2ms/step -
loss: 2.1125e-06
Epoch 33/40
354/354             1s 2ms/step -
loss: 1.2365e-06
Epoch 34/40
354/354             1s 2ms/step -
loss: 2.2970e-07
Epoch 35/40
354/354             1s 2ms/step -
loss: 2.1297e-06
```

```
Epoch 36/40
354/354              1s 2ms/step -
loss: 2.3426e-06
Epoch 37/40
354/354              1s 2ms/step -
loss: 3.7339e-07
Epoch 38/40
354/354              1s 2ms/step -
loss: 1.4670e-06
Epoch 39/40
354/354              1s 2ms/step -
loss: 3.9501e-07
Epoch 40/40
354/354              1s 1ms/step -
loss: 1.3417e-06
7/7            0s 10ms/step
Gelecekteki Fiyat Tahminleri:
[[14.37092  ]
 [14.36142  ]
 [14.363637 ]
 [14.398416 ]
 [14.356109 ]
 [14.352097 ]
 [14.351755 ]
 [14.352729 ]
 [14.477439 ]
 [14.356011 ]
 [14.356479 ]
 [14.362084 ]
 [14.362558 ]
 [14.365112 ]
 [14.368308 ]
 [14.370106 ]
 [14.370973 ]
 [14.370783 ]
 [14.371469 ]
 [14.377347 ]
 [14.379084 ]
 [14.387599 ]
 [14.388406 ]
 [14.388657 ]
 [14.389968 ]
 [14.391331 ]
 [14.400975 ]
 [14.40249  ]
 [14.408512 ]
 [14.40699  ]
 [14.407174 ]
```

```
[14.408674 ]
[14.409769 ]
[14.41008  ]
[14.409033 ]
[14.408136 ]
[14.410127 ]
[14.409014 ]
[14.403575 ]
[14.400427 ]
[14.414823 ]
[14.404262 ]
[14.419423 ]
[14.452425 ]
[14.455838 ]
[14.452818 ]
[14.453421 ]
[14.455773 ]
[14.457028 ]
[14.457166 ]
[14.45724  ]
[14.458043 ]
[14.458874 ]
[14.457919 ]
[14.484898 ]
[14.4625225]
[14.464486 ]
[14.463327 ]
[14.465419 ]
[14.466299 ]
[14.451173 ]
[14.44933  ]
[14.450191 ]
[14.452917 ]
[14.484275 ]
[14.482959 ]
[14.485825 ]
[14.486707 ]
[14.48096  ]
[14.479844 ]
[14.479516 ]
[14.479736 ]
[14.484439 ]
[14.479234 ]
[14.493387 ]
[14.479494 ]
[14.479502 ]
[14.479248 ]
[14.479599 ]
```

```
[14.483168 ]
[14.546634 ]
[14.482888 ]
[14.485623 ]
[14.486433 ]
[14.4826145]
[14.482985 ]
[14.489262 ]
[14.504268 ]
[14.502048 ]
[14.498484 ]
[14.498242 ]
[14.497439 ]
[14.497676 ]
[14.498901 ]
[14.499404 ]
[14.497835 ]
[14.498845 ]
[14.499626 ]
[14.499127 ]
[14.500405 ]
[14.499769 ]
[14.499552 ]
[14.502291 ]
[14.502725 ]
[14.507642 ]
[14.507665 ]
[14.506074 ]
[14.678766 ]
[14.508283 ]
[14.506937 ]
[14.511333 ]
[14.602653 ]
[14.534112 ]
[14.57933  ]
[14.587334 ]
[14.575702 ]
[14.653853 ]
[14.567198 ]
[14.520914 ]
[14.522655 ]
[14.520396 ]
[14.52156  ]
[14.521769 ]
[14.521181 ]
[14.521222 ]
[14.524241 ]
[14.523802 ]
```

```
[14.522474 ]
[14.522286 ]
[14.521803 ]
[14.521521 ]
[14.51876  ]
[14.518377 ]
[14.530593 ]
[14.53433  ]
[14.5607395]
[14.535752 ]
[14.536734 ]
[14.535107 ]
[14.535716 ]
[14.53538  ]
[14.534724 ]
[14.536947 ]
[14.5357065]
[14.535623 ]
[14.53747  ]
[14.534415 ]
[14.535145 ]
[14.535058 ]
[14.533175 ]
[14.533653 ]
[14.538173 ]
[14.537494 ]
[14.551138 ]
[14.580138 ]
[14.57268  ]
[14.572558 ]
[14.5722065]
[14.748278 ]
[14.776507 ]
[14.678849 ]
[14.584242 ]
[14.707205 ]
[14.576754 ]
[14.583414 ]
[14.575291 ]
[14.575097 ]
[14.577669 ]
[14.577321 ]
[14.579139 ]
[14.57815  ]
[14.576984 ]
[14.575767 ]
[14.573488 ]
[14.565437 ]
```

```
 [14.561133 ]
 [14.587017 ]
 [14.594309 ]
 [14.620886 ]
 [14.61996  ]
 [14.620692 ]
 [14.618488 ]
 [14.617715 ]
 [14.666649 ]
 [14.61486  ]
 [14.614808 ]
 [14.615795 ]
 [14.614849 ]
 [14.6158905]
 [14.61534  ]
 [14.615852 ]
 [14.616315 ]
 [14.614916 ]
 [14.616552 ]
 [14.615568 ]
 [14.615507 ]
 [14.615392 ]
 [14.645705 ]
 [14.654919 ]
 [14.637547 ]
 [14.633227 ]]
71/71            0s 815us/step
RMSE: 0.007185994284786264
MAE: 0.006429869016247972
```

[7]:
```python
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/ail_frx.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
```

```python
veri['close'].fillna(veri['close'].mean(), inplace=True)


veri = veri[['open', 'low', 'high', 'close']]

X = veri[['close', 'open', 'high']].values
y = veri['low'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
 ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=25, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)


tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)


y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/25
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
```

```
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
354/354             3s 2ms/step -
loss: 0.0234
Epoch 2/25
354/354             1s 2ms/step -
loss: 9.4186e-06
Epoch 3/25
354/354             1s 2ms/step -
loss: 7.2204e-06
Epoch 4/25
354/354             1s 2ms/step -
loss: 6.4341e-06
Epoch 5/25
354/354             1s 2ms/step -
loss: 6.9352e-06
Epoch 6/25
354/354             1s 2ms/step -
loss: 8.1289e-06
Epoch 7/25
354/354             1s 2ms/step -
loss: 6.6905e-06
Epoch 8/25
354/354             1s 2ms/step -
loss: 7.2243e-06
Epoch 9/25
354/354             1s 2ms/step -
loss: 6.0264e-06
Epoch 10/25
354/354             1s 2ms/step -
loss: 6.4109e-06
Epoch 11/25
354/354             1s 2ms/step -
loss: 7.2272e-06
Epoch 12/25
354/354             1s 2ms/step -
loss: 5.9459e-06
Epoch 13/25
354/354             1s 2ms/step -
loss: 6.7903e-06
Epoch 14/25
354/354             1s 2ms/step -
loss: 7.1161e-06
Epoch 15/25
354/354             1s 2ms/step -
loss: 7.8001e-06
Epoch 16/25
```

```
354/354             1s 2ms/step -
loss: 1.0129e-05
Epoch 17/25
354/354             1s 2ms/step -
loss: 8.5468e-06
Epoch 18/25
354/354             1s 2ms/step -
loss: 9.7683e-06
Epoch 19/25
354/354             1s 2ms/step -
loss: 6.2865e-06
Epoch 20/25
354/354             1s 2ms/step -
loss: 7.9681e-06
Epoch 21/25
354/354             1s 2ms/step -
loss: 6.0829e-06
Epoch 22/25
354/354             1s 2ms/step -
loss: 9.2659e-06
Epoch 23/25
354/354             1s 2ms/step -
loss: 8.6202e-06
Epoch 24/25
354/354             1s 2ms/step -
loss: 7.0496e-06
Epoch 25/25
354/354             1s 2ms/step -
loss: 7.1247e-06
7/7             0s 13ms/step
Gelecekteki Fiyat Tahminleri:
[[14.245492 ]
 [14.244752 ]
 [14.244617 ]
 [14.260704 ]
 [14.244955 ]
 [14.244984 ]
 [14.243921 ]
 [14.245583 ]
 [14.277243 ]
 [14.249397 ]
 [14.250165 ]
 [14.251717 ]
 [14.255325 ]
 [14.256014 ]
 [14.260753 ]
 [14.263605 ]
 [14.264639 ]
```

```
[14.263676 ]
[14.265212 ]
[14.221781 ]
[14.267796 ]
[14.2625   ]
[14.279585 ]
[14.282902 ]
[14.281944 ]
[14.282043 ]
[14.288142 ]
[14.293682 ]
[14.296921 ]
[14.298638 ]
[14.300764 ]
[14.30082  ]
[14.303091 ]
[14.302999 ]
[14.302261 ]
[14.301406 ]
[14.2520895]
[14.301395 ]
[14.29517  ]
[14.294153 ]
[14.297182 ]
[14.293831 ]
[14.300293 ]
[14.321107 ]
[14.343628 ]
[14.29673  ]
[14.297234 ]
[14.348319 ]
[14.302365 ]
[14.350209 ]
[14.347521 ]
[14.351366 ]
[14.35123  ]
[14.350514 ]
[14.357326 ]
[14.35381  ]
[14.355614 ]
[14.307557 ]
[14.357907 ]
[14.357887 ]
[14.343611 ]
[14.339692 ]
[14.33396  ]
[14.33783  ]
[14.362902 ]
```

```
[14.374081 ]
[14.375165 ]
[14.374935 ]
[14.373369 ]
[14.372967 ]
[14.372306 ]
[14.371986 ]
[14.373256 ]
[14.371862 ]
[14.376369 ]
[14.372649 ]
[14.3723755]
[14.372553 ]
[14.372229 ]
[14.373706 ]
[14.391855 ]
[14.37568  ]
[14.377093 ]
[14.378947 ]
[14.3754635]
[14.326288 ]
[14.378501 ]
[14.385924 ]
[14.393429 ]
[14.3408375]
[14.390923 ]
[14.38987  ]
[14.391066 ]
[14.340555 ]
[14.391903 ]
[14.391075 ]
[14.391309 ]
[14.340282 ]
[14.391819 ]
[14.391965 ]
[14.392368 ]
[14.392275 ]
[14.393826 ]
[14.396051 ]
[14.396786 ]
[14.400792 ]
[14.398482 ]
[14.443202 ]
[14.400525 ]
[14.391491 ]
[14.397818 ]
[14.418255 ]
[14.417663 ]
```

```
[14.441353 ]
[14.471614 ]
[14.4303   ]
[14.462458 ]
[14.423309 ]
[14.412408 ]
[14.413685 ]
[14.412933 ]
[14.414119 ]
[14.413848 ]
[14.414157 ]
[14.413817 ]
[14.415073 ]
[14.415735 ]
[14.415449 ]
[14.414965 ]
[14.414712 ]
[14.411559 ]
[14.410741 ]
[14.40756  ]
[14.359483 ]
[14.4208355]
[14.433473 ]
[14.428318 ]
[14.428353 ]
[14.427469 ]
[14.428238 ]
[14.427742 ]
[14.426618 ]
[14.370765 ]
[14.428617 ]
[14.428111 ]
[14.428742 ]
[14.370719 ]
[14.427348 ]
[14.427864 ]
[14.425084 ]
[14.426619 ]
[14.372066 ]
[14.42909  ]
[14.433265 ]
[14.442803 ]
[14.454555 ]
[14.46312  ]
[14.401499 ]
[14.531583 ]
[14.47449  ]
[14.49105  ]
```

```
 [14.472423 ]
 [14.502382 ]
 [14.466605 ]
 [14.468961 ]
 [14.46705  ]
 [14.46751  ]
 [14.468765 ]
 [14.469529 ]
 [14.470069 ]
 [14.46971  ]
 [14.468418 ]
 [14.466985 ]
 [14.4640045]
 [14.454479 ]
 [14.444294 ]
 [14.433194 ]
 [14.459487 ]
 [14.4884815]
 [14.510889 ]
 [14.509279 ]
 [14.510408 ]
 [14.440607 ]
 [14.518515 ]
 [14.506907 ]
 [14.506943 ]
 [14.506952 ]
 [14.506682 ]
 [14.507342 ]
 [14.507402 ]
 [14.507675 ]
 [14.508214 ]
 [14.507296 ]
 [14.5062275]
 [14.439    ]
 [14.507622 ]
 [14.507055 ]
 [14.514324 ]
 [14.4498625]
 [14.523104 ]
 [14.524891 ]]
71/71             0s 2ms/step
RMSE: 0.00261864428211394
MAE: 0.0016339600934450968
```

[8]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error


veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/ail_frx.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)


veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)



veri = veri[['open', 'low', 'high', 'close']]


X = veri[['close', 'open', 'high']].values
y = veri['close'].values


scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)


X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
  ↪test_size=0.2, random_state=42)


model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
  ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])


model.compile(optimizer='adam', loss='mean_squared_error')
```

```python
model.fit(X_scaled, y_scaled, epochs=25, batch_size=32, verbose=1)


gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)


tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)


y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/25
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
354/354              3s 2ms/step -
loss: 0.0459
Epoch 2/25
354/354              1s 2ms/step -
loss: 2.0051e-06
Epoch 3/25
354/354              1s 2ms/step -
loss: 3.4334e-07
Epoch 4/25
354/354              1s 2ms/step -
loss: 2.3760e-07
Epoch 5/25
354/354              1s 2ms/step -
loss: 1.7649e-07
Epoch 6/25
354/354              1s 2ms/step -
loss: 1.5046e-07
Epoch 7/25
354/354              1s 2ms/step -
loss: 1.8930e-07
```

```
Epoch 8/25
354/354            1s 2ms/step -
loss: 2.0901e-07
Epoch 9/25
354/354            1s 2ms/step -
loss: 1.6161e-07
Epoch 10/25
354/354            1s 2ms/step -
loss: 1.2695e-06
Epoch 11/25
354/354            1s 2ms/step -
loss: 2.6158e-07
Epoch 12/25
354/354            1s 2ms/step -
loss: 2.6054e-07
Epoch 13/25
354/354            1s 2ms/step -
loss: 3.0494e-06
Epoch 14/25
354/354            1s 2ms/step -
loss: 3.0851e-07
Epoch 15/25
354/354            1s 2ms/step -
loss: 3.0842e-07
Epoch 16/25
354/354            1s 2ms/step -
loss: 2.1421e-06
Epoch 17/25
354/354            1s 2ms/step -
loss: 5.3798e-08
Epoch 18/25
354/354            1s 2ms/step -
loss: 2.0522e-06
Epoch 19/25
354/354            1s 2ms/step -
loss: 8.8629e-07
Epoch 20/25
354/354            1s 2ms/step -
loss: 4.2907e-06
Epoch 21/25
354/354            1s 2ms/step -
loss: 1.7286e-06
Epoch 22/25
354/354            1s 2ms/step -
loss: 1.3526e-06
Epoch 23/25
354/354            1s 2ms/step -
loss: 1.3628e-05
```

```
Epoch 24/25
354/354            1s 2ms/step -
loss: 1.2759e-07
Epoch 25/25
354/354            1s 2ms/step -
loss: 1.4876e-07
7/7           0s 12ms/step
Gelecekteki Fiyat Tahminleri:
[[14.245327 ]
 [14.242742 ]
 [14.25019  ]
 [14.248877 ]
 [14.248027 ]
 [14.247018 ]
 [14.249766 ]
 [14.251161 ]
 [14.24859  ]
 [14.254551 ]
 [14.254421 ]
 [14.25936  ]
 [14.258871 ]
 [14.263444 ]
 [14.266914 ]
 [14.269058 ]
 [14.264736 ]
 [14.2668915]
 [14.268669 ]
 [ 9.937164 ]
 [14.254604 ]
 [14.282385 ]
 [14.287262 ]
 [14.286259 ]
 [14.2862215]
 [14.289896 ]
 [14.298087 ]
 [14.299222 ]
 [14.302453 ]
 [14.304588 ]
 [14.303806 ]
 [14.306985 ]
 [14.306754 ]
 [14.306972 ]
 [14.30582  ]
 [14.30519  ]
 [ 9.937077 ]
 [14.296471 ]
 [14.298399 ]
 [14.297739 ]
```

```
[14.29599  ]
[14.2936735]
[14.3098545]
[14.341825 ]
[14.346879 ]
[ 9.937212 ]
[ 9.937205 ]
[14.353946 ]
[ 9.93728  ]
[14.351195 ]
[14.354583 ]
[14.355508 ]
[14.354511 ]
[14.353707 ]
[14.3562   ]
[14.359046 ]
[14.358816 ]
[ 9.937238 ]
[14.363243 ]
[14.347876 ]
[14.340597 ]
[14.328246 ]
[14.344483 ]
[14.334461 ]
[14.376925 ]
[14.376521 ]
[14.374313 ]
[14.373195 ]
[14.374669 ]
[14.376617 ]
[14.375126 ]
[14.376198 ]
[14.376116 ]
[14.376818 ]
[14.376522 ]
[14.376175 ]
[14.376336 ]
[14.375905 ]
[14.376289 ]
[14.380788 ]
[14.3778305]
[14.379567 ]
[14.383424 ]
[14.378994 ]
[14.379214 ]
[ 9.937256 ]
[14.385197 ]
[14.390794 ]
```

```
[14.395933 ]
[ 9.937296 ]
[14.393457 ]
[14.394428 ]
[14.394912 ]
[ 9.937254 ]
[14.395618 ]
[14.395222 ]
[14.394082 ]
[ 9.937246 ]
[14.39561  ]
[14.396006 ]
[14.395873 ]
[14.39651  ]
[14.399791 ]
[14.398732 ]
[14.404558 ]
[14.403137 ]
[14.402344 ]
[14.394735 ]
[14.398231 ]
[14.400667 ]
[14.3994665]
[14.411894 ]
[14.425948 ]
[14.480757 ]
[14.417612 ]
[14.44209  ]
[14.41267  ]
[14.4156065]
[14.41619  ]
[14.4159155]
[14.417256 ]
[14.417037 ]
[14.417921 ]
[14.417455 ]
[14.4186535]
[14.420136 ]
[14.418953 ]
[14.418623 ]
[14.418788 ]
[14.415086 ]
[14.410498 ]
[14.4131155]
[14.394306 ]
[ 9.936829 ]
[14.430076 ]
[14.431245 ]
```

```
[14.430923 ]
[14.430892 ]
[14.429256 ]
[14.430428 ]
[14.430557 ]
[14.429913 ]
[ 9.937212 ]
[14.432052 ]
[14.432179 ]
[14.430978 ]
[ 9.937283 ]
[14.43122  ]
[14.432046 ]
[14.42941  ]
[14.4299   ]
[ 9.937174 ]
[14.431658 ]
[14.433568 ]
[14.467821 ]
[14.465318 ]
[14.466438 ]
[ 9.937229 ]
[14.682763 ]
[ 9.948185 ]
[14.46753  ]
[14.474325 ]
[14.464332 ]
[14.468856 ]
[14.469473 ]
[14.470972 ]
[14.470945 ]
[14.4731045]
[14.472972 ]
[14.473156 ]
[14.471646 ]
[14.471724 ]
[14.45963  ]
[14.450654 ]
[14.453568 ]
[14.411774 ]
[14.431269 ]
[14.485884 ]
[14.514363 ]
[14.512608 ]
[14.5113125]
[14.511154 ]
[ 9.937832 ]
[14.508514 ]
```

```
 [14.510259 ]
 [14.510249 ]
 [14.51153  ]
 [14.510822 ]
 [14.510973 ]
 [14.510752 ]
 [14.5117445]
 [14.510882 ]
 [14.510395 ]
 [14.510301 ]
 [ 9.937806 ]
 [14.510342 ]
 [14.510727 ]
 [14.500871 ]
 [ 9.936631 ]
 [14.527299 ]
 [14.529329 ]]
71/71             0s 1ms/step
RMSE: 0.00013405865215O5379
MAE: 0.00010264055692906284
```

crp_ail kısmı bu kodlar aracılığı ile yapıldı.

[9]:
```python
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/crp_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)


veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)



veri = veri[['open', 'low', 'high', 'close']]

X = veri[['close', 'low', 'high']].values
```

```python
y = veri['open'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
 ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])


model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=20, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)

tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/20
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
346/346              3s 2ms/step -
loss: 0.0134
Epoch 2/20
346/346              1s 2ms/step -
```

```
loss: 0.0032
Epoch 3/20
346/346              1s 2ms/step -
loss: 0.0035
Epoch 4/20
346/346              1s 2ms/step -
loss: 0.0031
Epoch 5/20
346/346              1s 2ms/step -
loss: 0.0032
Epoch 6/20
346/346              1s 2ms/step -
loss: 0.0031
Epoch 7/20
346/346              1s 2ms/step -
loss: 0.0034
Epoch 8/20
346/346              1s 2ms/step -
loss: 0.0037
Epoch 9/20
346/346              1s 2ms/step -
loss: 0.0037
Epoch 10/20
346/346              1s 2ms/step -
loss: 0.0034
Epoch 11/20
346/346              1s 2ms/step -
loss: 0.0034
Epoch 12/20
346/346              1s 2ms/step -
loss: 0.0033
Epoch 13/20
346/346              1s 2ms/step -
loss: 0.0033
Epoch 14/20
346/346              1s 2ms/step -
loss: 0.0034
Epoch 15/20
346/346              1s 2ms/step -
loss: 0.0031
Epoch 16/20
346/346              1s 2ms/step -
loss: 0.0031
Epoch 17/20
346/346              1s 2ms/step -
loss: 0.0032
Epoch 18/20
346/346              1s 2ms/step -
```

```
loss: 0.0031
Epoch 19/20
346/346             1s 2ms/step -
loss: 0.0038
Epoch 20/20
346/346             1s 2ms/step -
loss: 0.0034
7/7             0s 13ms/step
Gelecekteki Fiyat Tahminleri:
[[302.71118]
 [302.5584 ]
 [302.4678 ]
 [302.44077]
 [302.36328]
 [302.29083]
 [302.13602]
 [302.15274]
 [302.2982 ]
 [302.2537 ]
 [302.3346 ]
 [302.31302]
 [303.16376]
 [303.55057]
 [304.47092]
 [304.0437 ]
 [304.81683]
 [304.90115]
 [305.18546]
 [302.97073]
 [305.13684]
 [305.2045 ]
 [304.964  ]
 [305.27948]
 [305.54535]
 [306.32504]
 [306.28555]
 [304.0476 ]
 [306.35812]
 [306.2134 ]
 [304.45078]
 [306.1448 ]
 [306.26617]
 [306.89972]
 [305.97653]
 [305.97095]
 [306.34097]
 [306.4167 ]
 [306.54343]
```

```
[306.61942]
[306.6148 ]
[306.69037]
[306.83087]
[306.9505 ]
[306.69928]
[306.29266]
[306.2607 ]
[306.3829 ]
[306.65143]
[306.79385]
[307.41068]
[307.79355]
[307.8712 ]
[307.78925]
[307.74637]
[308.3831 ]
[308.02277]
[308.4479 ]
[308.96317]
[310.4981 ]
[309.34644]
[308.50623]
[308.97342]
[309.15387]
[309.2964 ]
[307.22144]
[309.99265]
[309.77966]
[309.20184]
[309.2088 ]
[309.44058]
[309.29105]
[309.20477]
[309.43918]
[309.37573]
[309.23865]
[309.22748]
[309.116  ]
[309.193  ]
[309.42392]
[309.35312]
[309.77744]
[309.61346]
[309.5499 ]
[309.26675]
[309.6984 ]
[309.6406 ]
```

```
[309.71375]
[309.75308]
[309.70328]
[309.5902 ]
[309.48566]
[309.10495]
[309.14084]
[308.9551 ]
[308.77463]
[308.8638 ]
[309.01093]
[309.07498]
[308.91916]
[308.54132]
[308.381  ]
[307.1792 ]
[306.9235 ]
[307.27612]
[306.8153 ]
[306.03375]
[305.95984]
[306.40826]
[306.49313]
[306.5877 ]
[306.3989 ]
[306.54538]
[306.52167]
[306.67468]
[306.61118]
[306.5244 ]
[306.67554]
[306.71628]
[306.68274]
[304.6995 ]
[306.57925]
[306.86758]
[307.1107 ]
[307.29825]
[307.63928]
[308.25906]
[308.36102]
[308.9467 ]
[308.7693 ]
[308.5441 ]
[308.62326]
[308.66455]
[308.7685 ]
[308.75977]
```

```
[308.7375 ]
[308.58823]
[308.28275]
[308.09772]
[307.89673]
[307.81332]
[307.93347]
[308.05444]
[307.99612]
[307.92725]
[307.46918]
[307.13922]
[306.85098]
[306.43335]
[306.53635]
[306.95746]
[306.89584]
[307.00403]
[307.09268]
[306.97614]
[306.9731 ]
[306.9168 ]
[306.97253]
[307.2831 ]
[307.0247 ]
[305.1517 ]
[307.337  ]
[307.59897]
[307.8759 ]
[307.83136]
[307.71957]
[308.08875]
[308.14465]
[307.7586 ]
[307.09882]
[306.7474 ]
[306.908  ]
[306.83752]
[306.46112]
[306.5236 ]
[306.45364]
[306.31418]
[305.9411 ]
[306.12387]
[306.23428]
[303.9462 ]
[305.00873]
[305.02844]
```

```
 [305.03036]
 [305.12247]
 [305.38367]
 [305.713  ]
 [306.0129 ]
 [306.27936]
 [306.70395]
 [306.86816]
 [304.54895]
 [306.6244 ]
 [306.44397]
 [306.6805 ]
 [306.70712]
 [306.74442]
 [306.70386]
 [306.75064]
 [306.89584]
 [306.89642]]
70/70            0s 1ms/step
RMSE: 0.06217185445865172
MAE: 0.027066357016339693
```

[10]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error


veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/crp_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)


veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)



veri = veri[['open', 'low', 'high', 'close']]
```

```
X = veri[['close', 'low', 'high']].values
y = veri['high'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
  ↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
  ↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=20, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)


tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/20

C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

346/346             3s 2ms/step -
loss: 0.0247
Epoch 2/20
```

```
346/346             1s 2ms/step -
loss: 1.3462e-05
Epoch 3/20
346/346             1s 2ms/step -
loss: 8.8164e-06
Epoch 4/20
346/346             1s 2ms/step -
loss: 7.0121e-06
Epoch 5/20
346/346             1s 2ms/step -
loss: 5.7155e-06
Epoch 6/20
346/346             1s 2ms/step -
loss: 4.9056e-06
Epoch 7/20
346/346             1s 2ms/step -
loss: 4.4036e-06
Epoch 8/20
346/346             1s 2ms/step -
loss: 3.8157e-06
Epoch 9/20
346/346             1s 2ms/step -
loss: 2.6242e-06
Epoch 10/20
346/346             1s 2ms/step -
loss: 1.9687e-06
Epoch 11/20
346/346             1s 1ms/step -
loss: 1.8634e-06
Epoch 12/20
346/346             1s 2ms/step -
loss: 9.4455e-07
Epoch 13/20
346/346             1s 2ms/step -
loss: 8.2270e-07
Epoch 14/20
346/346             1s 2ms/step -
loss: 6.2134e-07
Epoch 15/20
346/346             1s 2ms/step -
loss: 6.2863e-07
Epoch 16/20
346/346             1s 2ms/step -
loss: 3.6657e-07
Epoch 17/20
346/346             1s 2ms/step -
loss: 3.4710e-07
Epoch 18/20
```

```
346/346             1s 2ms/step -
loss: 4.0628e-07
Epoch 19/20
346/346             1s 2ms/step -
loss: 4.6238e-07
Epoch 20/20
346/346             1s 2ms/step -
loss: 6.8817e-07
7/7             0s 10ms/step
Gelecekteki Fiyat Tahminleri:
[[304.3568 ]
 [304.2312 ]
 [303.95346]
 [304.00488]
 [303.95926]
 [303.9359 ]
 [303.76617]
 [303.7061 ]
 [303.93616]
 [303.8753 ]
 [304.1085 ]
 [304.0291 ]
 [304.94028]
 [305.5903 ]
 [306.9397 ]
 [306.43857]
 [307.04343]
 [307.06985]
 [307.3823 ]
 [306.73523]
 [306.8234 ]
 [307.03   ]
 [306.926  ]
 [307.36792]
 [307.43005]
 [308.7068 ]
 [308.32184]
 [307.9275 ]
 [308.4707 ]
 [308.1869 ]
 [308.28555]
 [308.17746]
 [308.32953]
 [309.176  ]
 [308.44156]
 [308.28006]
 [308.33362]
 [308.3984 ]
```

```
[308.621  ]
[308.57547]
[308.52707]
[308.51328]
[308.8729 ]
[309.011  ]
[308.75626]
[308.27936]
[308.31964]
[308.2786 ]
[308.79462]
[308.775  ]
[309.73816]
[310.07355]
[310.0033 ]
[309.88306]
[309.8304 ]
[312.07095]
[310.58536]
[310.95822]
[311.45844]
[313.49496]
[312.04874]
[311.03067]
[311.10013]
[311.19986]
[311.56427]
[311.43585]
[312.6367 ]
[312.29776]
[311.41464]
[311.4053 ]
[311.58856]
[311.58893]
[311.36682]
[311.73373]
[311.77542]
[311.57468]
[311.46387]
[311.2902 ]
[311.45157]
[312.07242]
[311.81332]
[312.16263]
[312.0145 ]
[311.93353]
[311.49432]
[311.94315]
```

```
[311.80804]
[311.7964 ]
[312.0403 ]
[312.004  ]
[311.8173 ]
[311.76154]
[311.22906]
[311.2589 ]
[311.23596]
[310.9508 ]
[310.99133]
[311.1368 ]
[311.22656]
[311.02505]
[310.8622 ]
[311.59   ]
[309.86197]
[309.1935 ]
[309.42297]
[309.31348]
[308.00922]
[308.11694]
[308.35953]
[308.27847]
[308.70535]
[308.38672]
[308.57278]
[308.4464 ]
[308.72784]
[308.60416]
[308.46307]
[308.59274]
[308.70453]
[308.6192 ]
[308.57367]
[308.53537]
[309.02322]
[309.1038 ]
[309.40442]
[310.1368 ]
[310.57056]
[310.59778]
[311.33078]
[310.86783]
[310.72577]
[310.60236]
[310.7178 ]
[310.82755]
```

```
[310.87854]
[310.9301 ]
[310.7136 ]
[310.2956 ]
[310.21292]
[309.96146]
[309.79153]
[309.9463 ]
[310.21274]
[310.05893]
[310.05637]
[309.9374 ]
[309.50888]
[309.04187]
[308.8031 ]
[308.54346]
[309.1988 ]
[308.8449 ]
[308.99786]
[309.0576 ]
[308.80096]
[308.8546 ]
[308.81122]
[308.9498 ]
[309.35278]
[308.93237]
[309.0574 ]
[309.28033]
[309.81824]
[309.94968]
[309.9232 ]
[309.84882]
[310.3161 ]
[310.30566]
[309.83087]
[309.53638]
[308.9746 ]
[309.10458]
[308.99442]
[308.58344]
[308.54028]
[308.42593]
[308.26086]
[307.94797]
[308.065  ]
[308.1504 ]
[307.8904 ]
[306.8049 ]
```

```
 [306.8346 ]
 [306.83466]
 [306.947  ]
 [307.38278]
 [307.69827]
 [307.91608]
 [308.28583]
 [308.6862 ]
 [309.00934]
 [308.50143]
 [308.6846 ]
 [308.60498]
 [308.67014]
 [308.6021 ]
 [308.64764]
 [308.56342]
 [308.55823]
 [308.86697]
 [308.81738]]
70/70             0s 1ms/step
RMSE: 0.0007563394079618845
MAE: 0.0007179458762785808
```

```
[11]: import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/crp_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)


veri = veri[['open', 'low', 'high', 'close']]

X = veri[['close', 'low', 'high']].values
y = veri['low'].values
```

```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=20, batch_size=32, verbose=1)

gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)

tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/20
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
346/346              3s 2ms/step -
loss: 0.0365
Epoch 2/20
346/346              1s 2ms/step -
loss: 2.6020e-05
Epoch 3/20
```

```
346/346             1s 2ms/step -
loss: 1.5432e-05
Epoch 4/20
346/346             1s 2ms/step -
loss: 9.7915e-06
Epoch 5/20
346/346             1s 2ms/step -
loss: 7.0072e-06
Epoch 6/20
346/346             1s 2ms/step -
loss: 3.8594e-06
Epoch 7/20
346/346             1s 2ms/step -
loss: 2.3884e-06
Epoch 8/20
346/346             1s 2ms/step -
loss: 1.7009e-06
Epoch 9/20
346/346             1s 2ms/step -
loss: 8.6841e-07
Epoch 10/20
346/346             1s 2ms/step -
loss: 5.1009e-07
Epoch 11/20
346/346             1s 2ms/step -
loss: 2.6750e-07
Epoch 12/20
346/346             1s 2ms/step -
loss: 2.7685e-07
Epoch 13/20
346/346             1s 1ms/step -
loss: 1.6991e-07
Epoch 14/20
346/346             1s 2ms/step -
loss: 4.7751e-07
Epoch 15/20
346/346             1s 2ms/step -
loss: 2.9043e-07
Epoch 16/20
346/346             1s 2ms/step -
loss: 2.0052e-06
Epoch 17/20
346/346             1s 2ms/step -
loss: 1.5541e-07
Epoch 18/20
346/346             1s 2ms/step -
loss: 8.0056e-07
Epoch 19/20
```

```
346/346             1s 2ms/step -
loss: 9.1724e-07
Epoch 20/20
346/346             1s 2ms/step -
loss: 3.8163e-06
7/7             0s 13ms/step
Gelecekteki Fiyat Tahminleri:
[[304.11304]
 [303.90225]
 [304.025  ]
 [303.89487]
 [303.76694]
 [303.625  ]
 [303.45346]
 [303.56934]
 [303.64346]
 [303.60175]
 [303.51993]
 [303.5633 ]
 [304.476  ]
 [304.5941 ]
 [305.14713]
 [304.74075]
 [305.84985]
 [306.03748]
 [306.33472]
 [306.2368 ]
 [306.9035 ]
 [306.81   ]
 [306.36255]
 [306.56705]
 [307.13483]
 [307.43054]
 [307.82645]
 [307.4503 ]
 [307.81616]
 [307.804  ]
 [308.00848]
 [307.6551 ]
 [307.75253]
 [308.24673]
 [306.92526]
 [307.10263]
 [307.93127]
 [308.04108]
 [308.06244]
 [308.3104 ]
 [308.35248]
```

```
[308.55524]
[308.44434]
[308.57025]
[308.28482]
[307.89273]
[307.75534]
[308.10394]
[308.1062 ]
[308.4777 ]
[308.77548]
[309.2785 ]
[309.5574 ]
[309.50687]
[309.4768 ]
[308.22684]
[309.1971 ]
[309.74707]
[310.36966]
[311.5539 ]
[310.59753]
[309.80298]
[310.85306]
[311.1982 ]
[311.0679 ]
[310.73196]
[311.3882 ]
[311.32428]
[311.0214 ]
[311.04105]
[311.39685]
[311.02124]
[311.08612]
[311.1852 ]
[310.99997]
[310.90045]
[311.02173]
[310.95938]
[310.95502]
[310.72324]
[310.8769 ]
[311.47116]
[311.26965]
[311.21948]
[311.0672 ]
[311.5627 ]
[311.61816]
[311.83496]
[311.5762 ]
```

```
[311.50366]
[311.46655]
[311.27777]
[311.0143 ]
[311.06976]
[310.64566]
[310.56128]
[310.72177]
[310.90012]
[310.94623]
[310.8171 ]
[310.11542]
[308.81384]
[308.0774 ]
[308.26938]
[308.83408]
[307.8906 ]
[307.6029 ]
[307.273  ]
[308.0634 ]
[308.3706 ]
[308.07162]
[308.0126 ]
[308.1302 ]
[308.2314 ]
[308.24902]
[308.2524 ]
[308.21448]
[308.41965]
[308.38364]
[308.4057 ]
[308.27124]
[308.26187]
[308.34186]
[308.8332 ]
[308.92056]
[308.83658]
[309.78842]
[309.99564]
[310.51456]
[310.6529 ]
[310.28882]
[310.64233]
[310.58368]
[310.69937]
[310.61032]
[310.49878]
[310.41074]
```

```
[310.18637]
[309.84976]
[309.67056]
[309.67783]
[309.77536]
[309.73904]
[309.78467]
[309.62872]
[308.68207]
[308.40796]
[308.2839 ]
[307.58026]
[308.147  ]
[308.35754]
[308.63992]
[308.70618]
[308.85422]
[308.8878 ]
[308.81586]
[308.73184]
[308.69168]
[308.95145]
[308.84192]
[308.75793]
[309.16498]
[309.1191 ]
[309.63358]
[309.56543]
[309.3761 ]
[309.68723]
[309.8476 ]
[309.5023 ]
[308.2777 ]
[308.1218 ]
[308.354  ]
[308.32346]
[307.9219 ]
[308.1172 ]
[308.09802]
[307.96634]
[307.44632]
[307.74072]
[307.90448]
[307.254  ]
[306.61978]
[306.62848]
[306.63416]
[306.7158 ]
```

```
 [306.79684]
 [307.2048 ]
 [307.6617 ]
 [307.84015]
 [308.36923]
 [308.37888]
 [307.999  ]
 [308.18784]
 [307.8396 ]
 [308.33655]
 [308.48785]
 [308.51935]
 [308.52438]
 [308.6449 ]
 [308.61313]
 [308.6723 ]]
70/70             0s 2ms/step
RMSE: 0.00037814870053846196
MAE: 0.0003494904348502573
```

[12]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/crp_ail.csv")

veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['open'].fillna(veri['open'].mean(), inplace=True)
veri['high'].fillna(veri['high'].mean(), inplace=True)
veri['low'].fillna(veri['low'].mean(), inplace=True)
veri['close'].fillna(veri['close'].mean(), inplace=True)


veri = veri[['open', 'low', 'high', 'close']]

X = veri[['close', 'low', 'high']].values
y = veri['close'].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

```
y_scaled = scaler.fit_transform(y.reshape(-1,1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
↪test_size=0.2, random_state=42)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.
↪shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_scaled, y_scaled, epochs=20, batch_size=32, verbose=1)


gelecek_veri = X_scaled[-201:]
tahminler = model.predict(gelecek_veri)


tahminler = scaler.inverse_transform(tahminler)

print("Gelecekteki Fiyat Tahminleri:")
print(tahminler)


y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
print("RMSE:", rmse)
print("MAE:", mae)
```

```
Epoch 1/20

C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

346/346               3s 2ms/step -
loss: 0.0369
Epoch 2/20
346/346               1s 2ms/step -
loss: 2.9132e-06
Epoch 3/20
```

```
346/346              1s 2ms/step -
loss: 1.5659e-06
Epoch 4/20
346/346              1s 2ms/step -
loss: 1.0600e-06
Epoch 5/20
346/346              1s 2ms/step -
loss: 5.4161e-07
Epoch 6/20
346/346              1s 2ms/step -
loss: 5.9794e-07
Epoch 7/20
346/346              1s 2ms/step -
loss: 3.3842e-07
Epoch 8/20
346/346              1s 2ms/step -
loss: 2.8209e-07
Epoch 9/20
346/346              1s 2ms/step -
loss: 2.5849e-07
Epoch 10/20
346/346              1s 2ms/step -
loss: 8.1465e-07
Epoch 11/20
346/346              1s 2ms/step -
loss: 4.4596e-07
Epoch 12/20
346/346              1s 2ms/step -
loss: 2.6183e-07
Epoch 13/20
346/346              1s 2ms/step -
loss: 1.7326e-06
Epoch 14/20
346/346              1s 2ms/step -
loss: 4.6759e-07
Epoch 15/20
346/346              1s 2ms/step -
loss: 6.7310e-07
Epoch 16/20
346/346              1s 2ms/step -
loss: 2.7283e-06
Epoch 17/20
346/346              1s 2ms/step -
loss: 7.9965e-07
Epoch 18/20
346/346              1s 2ms/step -
loss: 4.5928e-06
Epoch 19/20
```

```
346/346             1s 2ms/step -
loss: 8.8650e-08
Epoch 20/20
346/346             1s 2ms/step -
loss: 9.8152e-08
7/7             0s 12ms/step
Gelecekteki Fiyat Tahminleri:
[[304.3181 ]
 [304.09055]
 [304.08466]
 [304.09818]
 [303.93784]
 [303.7332 ]
 [303.8583 ]
 [303.83618]
 [303.7076 ]
 [303.9915 ]
 [303.67664]
 [303.75266]
 [304.98486]
 [305.72015]
 [306.5244 ]
 [305.97   ]
 [306.9756 ]
 [306.48846]
 [306.83017]
 [270.4711 ]
 [306.97546]
 [307.0773 ]
 [306.70297]
 [307.33926]
 [307.48883]
 [308.42188]
 [307.8853 ]
 [270.49118]
 [307.99478]
 [308.33643]
 [270.5042 ]
 [308.10315]
 [308.46143]
 [308.62732]
 [307.83505]
 [308.1345 ]
 [308.42087]
 [308.25003]
 [308.67337]
 [308.46777]
 [308.68784]
```

```
[308.64746]
[308.94574]
[308.7848 ]
[308.3588 ]
[308.00705]
[308.33017]
[308.34906]
[308.84158]
[308.8605 ]
[309.40045]
[309.96286]
[309.88306]
[309.8574 ]
[309.53918]
[310.28006]
[310.21667]
[311.11166]
[311.6343 ]
[312.27316]
[310.73654]
[311.0019 ]
[311.15805]
[311.33646]
[311.1309 ]
[270.59695]
[312.48834]
[311.5642 ]
[311.16238]
[311.46216]
[311.63043]
[311.23618]
[311.4582 ]
[311.85873]
[311.1429 ]
[311.59235]
[311.2065 ]
[311.3807 ]
[311.13202]
[311.78015]
[311.80856]
[312.13736]
[311.64163]
[311.48035]
[311.63614]
[311.73022]
[311.92493]
[311.90607]
[311.72437]
```

```
[311.58426]
[311.92844]
[311.40256]
[311.16995]
[311.27573]
[310.97693]
[310.8659 ]
[311.10202]
[311.22098]
[311.1011 ]
[311.03818]
[310.38882]
[310.10413]
[308.71478]
[309.08826]
[309.50565]
[307.94894]
[307.74228]
[308.2841 ]
[308.42572]
[308.43594]
[308.48773]
[308.2328 ]
[308.55966]
[308.46915]
[308.72684]
[308.54593]
[308.57318]
[308.68372]
[308.58627]
[308.64365]
[270.50803]
[308.47192]
[309.15176]
[309.23245]
[309.18222]
[309.52396]
[310.34943]
[310.62143]
[310.74637]
[310.79663]
[310.46277]
[310.75156]
[310.77097]
[310.89398]
[311.00046]
[310.77863]
[310.4962 ]
```

```
[310.35623]
[309.92392]
[309.9151 ]
[309.89694]
[310.04117]
[310.1017 ]
[310.18365]
[309.87756]
[309.05   ]
[309.0154 ]
[308.95743]
[308.26825]
[308.47278]
[308.69342]
[308.86826]
[309.15646]
[308.9687 ]
[308.95377]
[308.8919 ]
[308.90305]
[309.05948]
[309.04977]
[309.01315]
[270.51752]
[309.30923]
[309.97556]
[309.93774]
[309.67636]
[309.961  ]
[310.3595 ]
[309.99216]
[309.6663 ]
[308.9334 ]
[308.70566]
[308.67224]
[308.49683]
[308.20337]
[308.56253]
[308.22025]
[308.0702 ]
[307.95   ]
[308.18375]
[308.18716]
[270.4846 ]
[306.76868]
[306.85995]
[306.8229 ]
[306.96286]
```

```
 [307.5458 ]
 [307.73682]
 [307.92972]
 [308.41483]
 [308.8231 ]
 [308.5313 ]
 [270.49933]
 [308.45694]
 [308.66806]
 [308.6646 ]
 [308.5927 ]
 [308.69977]
 [308.7057 ]
 [308.71057]
 [308.85242]
 [308.9755 ]]
70/70             0s 2ms/step
RMSE: 0.0006083523823731347
MAE: 0.0005342891759797481
```

bk_frx kısmı bu kodlar aracılığı ile yapıldı.

[13]:
```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/bk_frx.csv")
veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['FuturePrice'] = veri['open'].shift(-1)

veri.dropna(inplace=True)

X = veri[['open', 'low', 'high', 'close']].tail(4886)
y = veri['FuturePrice'].tail(4886)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1)).reshape(-1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
↪test_size=0.2, random_state=42)


rf_model = RandomForestRegressor()
rf_model.fit(X, y)


gelecek_veri = X.tail(89)[::-1]
rf_tahminler = rf_model.predict(gelecek_veri)


rf_mse = mean_squared_error(y[-88:], rf_tahminler[:-1])
rf_rmse = np.sqrt(rf_mse)

print("Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:")
np.set_printoptions(precision=7)
print(rf_tahminler)
print("RMSE Değeri:", rf_rmse)
```

```
Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:
[20140.377913  20190.1059696 20200.236113  20240.2605609 20321.4754565
 20327.9324696 20231.0618348 20380.6681609 20455.706713  20476.9304478
 20463.8101826 20519.492213  20524.4977174 20549.4236174 20549.1772826
 20518.0690478 20509.1887826 20620.0689304 20578.7358435 20582.468887
 20658.5480826 20611.009387  20659.2504478 20752.3343478 20949.4086261
 20955.8248522 20887.0930696 20998.0409652 21087.659587  21122.7266913
 21061.8165565 21065.8180565 21064.0722826 21243.4083826 21366.5034174
 21331.9361261 21585.6351435 21128.3430783 20960.5547565 20924.9161739
 20908.5587609 20942.2326565 20825.2058913 20721.5878957 20710.1666261
 20464.5950435 20291.0529957 20223.084413  20163.3969652 20339.6742739
 20397.5843783 20438.5476174 20385.1766174 20395.8754739 20465.9829522
 20396.8534391 20404.4619348 20437.4544348 20549.5563478 20637.1194913
 20611.4661087 20635.119087  20572.3398957 20545.5882609 20539.0858522
 20933.4844304 20902.1920783 21123.9788739 21164.477787  21232.2929
 21244.4287478 21125.5951435 21189.3725783 21009.707187  21059.3778652
 20979.1215783 20943.9770478 21103.1464174 21243.7607174 21163.9759
 21168.0518304 21325.4549174 21384.3312348 21301.5182739 21546.664013
 21546.7953609 21658.8191565 21620.8082478 21585.8598174]
RMSE Değeri: 755.593732240723
```

[14]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
```

```python
veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/bk_frx.csv")
veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)


veri['FuturePrice'] = veri['high'].shift(-1)


veri.dropna(inplace=True)


X = veri[['open', 'low', 'high', 'close']].tail(4886)
y = veri['FuturePrice'].tail(4886)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)

rf_model = RandomForestRegressor()
rf_model.fit(X, y)

gelecek_veri = X.tail(89)[::-1]
rf_tahminler = rf_model.predict(gelecek_veri)

rf_mse = mean_squared_error(y[-88:], rf_tahminler[:-1])
rf_rmse = np.sqrt(rf_mse)

print("Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:")
np.set_printoptions(precision=7)
print(rf_tahminler)
print("RMSE Değeri:", rf_rmse)
```

```
Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:
[20151.4565261 20225.8049261 20243.7560261 20295.4697391 20357.227413
 20385.3243261 20366.6554217 20403.5231435 20367.456413  20464.727313
 20475.1763783 20522.1703261 20519.073513  20529.4518348 20483.1738478
 20548.1472391 20603.2859043 20577.7106391 20564.5984391 20589.9935043
 20664.1672348 20590.4345565 20607.800813  20755.3235478 20698.3918696
 20693.6433    20772.6704478 20964.967813  20963.2992348 20898.903313
 21131.7666478 21139.7885435 21107.4140696 21087.9706739 21076.4906435
 21373.1450087 21392.2588826 21602.0374913 21097.6911043 21158.5127522
 21080.2674696 21061.2718087 20982.3027304 20948.5813652 20999.264113
 20948.5373522 20864.6305174 20804.176413  20734.3585913 20511.3820609
 20451.2531    20241.4042087 20177.9871739 20402.6901217 20415.1128609
 20472.8635043 20434.1188913 20426.1988478 20494.9251    20492.2776348
 20464.8943783 20567.6989391 20661.6928261 20669.6140217 20657.1712348
 20632.3649304 20612.705387  20549.9049957 20805.6772348 20942.0169696
 21091.9694087 20989.9384174 21155.9701348 21178.9933348 21255.9417304
 21273.0365348 21177.1491652 21200.3836739 21155.6862391 21091.6214522
 21089.4786391 21095.3839913 20958.4156304 21132.0198957 21207.7163391
 21173.2870783 21426.0171174 21389.2023957 21327.4067   ]
RMSE Değeri: 653.8977609132194
```

[15]:

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/bk_frx.csv")
veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['FuturePrice'] = veri['low'].shift(-1)

veri.dropna(inplace=True)

X = veri[['open', 'low', 'high', 'close']].tail(4886)
y = veri['FuturePrice'].tail(4886)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)

rf_model = RandomForestRegressor()
rf_model.fit(X, y)

gelecek_veri = X.tail(89)[::-1]
rf_tahminler = rf_model.predict(gelecek_veri)

rf_mse = mean_squared_error(y[-88:], rf_tahminler[:-1])
```

```
rf_rmse = np.sqrt(rf_mse)

print("Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:")
np.set_printoptions(precision=7)
print(rf_tahminler)
print("RMSE Değeri:", rf_rmse)
```

```
Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:
[20079.6389913 20125.6234043 20164.3304565 20211.5209652 20232.4790087
 20292.4473565 20238.6922217 20325.6707522 20303.2667304 20429.2191565
 20409.0980522 20411.1703522 20420.9949391 20505.5774087 20429.9118435
 20484.321413  20435.8628217 20465.9126696 20518.6098783 20558.5596652
 20535.3564957 20602.1438696 20625.9201304 20630.1755696 20756.8573217
 20953.3949174 20878.3989957 20868.0948739 21008.7312957 21061.5976435
 21035.767287  21009.0541348 21235.1530565 21295.9373739 21043.9871261
 21089.0624739 20996.5940652 20935.2389609 20898.9125304 20825.0752348
 20881.8801739 20942.7444522 20836.4577913 20667.0661043 20270.0193696
 20173.8752957 20153.6928957 20332.5669739 20363.2424522 20349.7180043
 20325.5294957 20332.2517391 20328.782313  20393.463513  20505.176913
 20544.7665304 20563.388887  20580.542913  20540.3537043 20479.2331826
 20522.017087  20645.213513  20860.8930957 20872.6602478 20944.6817174
 20921.470713  21140.7215739 21143.2844696 21142.0668522 21125.3043348
 20983.143587  20973.1913391 20942.9055261 20930.4201087 21157.0363565
 21310.7824435 21317.0903652 21390.4098739 21493.6176957 21476.4933957
 21535.6934739 21592.0006739 21548.8319435 21532.7863087 21405.0682913
 21505.138513  21516.162513  21533.9241957 21418.6833   ]
RMSE Değeri: 802.0540051505249
```

[16]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np


veri = pd.read_csv(r"C:/Users/ASUS/Desktop/AI CLUB DATATHON/CSV/bk_frx.csv")
veri['open'] = veri['open'].str.replace('"', '').astype(float)
veri['high'] = veri['high'].str.replace('"', '').astype(float)
veri['low'] = veri['low'].str.replace('"', '').astype(float)
veri['close'] = veri['close'].str.replace('"', '').astype(float)

veri['FuturePrice'] = veri['close'].shift(-1)

veri.dropna(inplace=True)

X = veri[['open', 'low', 'high', 'close']].tail(4886)
```

```python
y = veri['FuturePrice'].tail(4886)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1)).reshape(-1)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,␣
 ↪test_size=0.2, random_state=42)

rf_model = RandomForestRegressor()
rf_model.fit(X, y)

gelecek_veri = X.tail(89)[::-1]
rf_tahminler = rf_model.predict(gelecek_veri)

rf_mse = mean_squared_error(y[-88:], rf_tahminler[:-1])
rf_rmse = np.sqrt(rf_mse)

print("Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:")
np.set_printoptions(precision=7)
print(rf_tahminler)
print("RMSE Değeri:", rf_rmse)
```

```
Random Forest Regressor ile Gelecekteki Fiyat Tahminleri:
[20095.2958826 20167.8003435 20197.4672087 20225.9653087 20256.1234609
 20329.301713  20330.298113  20273.0673304 20359.5748522 20361.8994783
 20459.1344304 20449.7552739 20481.4326826 20456.4867348 20471.9081217
 20470.2250261 20535.6602087 20566.3861522 20512.3369826 20523.103587
 20536.2121    20546.0002783 20576.7034087 20566.7244304 20652.2927
 20666.1579609 20635.4935435 20678.7569826 20828.561713  20961.3684217
 20892.6306478 20895.3182087 21011.4428217 21084.8330739 21096.2363696
 21056.7769478 21059.7585435 21304.3867261 21339.7204435 21362.8671565
 21423.552387  21081.2873739 21113.4975478 21040.7610391 21001.9961478
 20932.5456391 20915.7672217 20897.2068522 20949.7234    20860.2944261
 20797.341487  20712.1829304 20451.852     20398.9992478 20217.2509565
 20164.9818957 20271.0466478 20357.1907739 20394.8138609 20397.7060478
 20376.6461522 20416.4175826 20461.8574783 20470.2192652 20413.4707826
 20591.3032957 20575.6079217 20612.1366739 20611.3937522 20613.8617087
 20552.2662609 20542.2040957 20684.9347087 20886.8243826 20936.2581739
 21018.7976087 20933.9584348 21095.4077261 21140.217613  21176.8698783
 21220.3923261 21109.7949217 21143.0966652 21125.1918826 21059.5281087
 21069.3022304 20994.3666826 20946.4855609 20966.0842696]
RMSE Değeri: 549.0391426919456
```