import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

import seaborn as sns

from datetime import datetime

import calendar

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

df=pd.read\_csv('/content/Final-50-stocks.csv')

df.sample(5)

df.isna().sum()

idx=np.where(df.isna().sum()>0)[0]     #finding index of columns where nan-values are more than 0

cols=df.iloc[:,idx].columns

cols

from sklearn.impute import KNNImputer

imputer=KNNImputer(n\_neighbors=2)

df[cols]=imputer.fit\_transform(df[cols])         #fitting impute nan values with two neighbouring values

df[cols]=np.round(df[cols],2)

df.DATE=df.DATE.apply(pd.to\_datetime)

df['year']=df.DATE.apply(lambda x:datetime.date(x).year)

df['month']=df.DATE.apply(lambda x:datetime.date(x).month)

df['day']=df.DATE.apply(lambda x:datetime.date(x).day)

df['day\_num']=df.DATE.apply(lambda x:datetime.date(x).weekday())

df['week\_num']=df.DATE.apply(lambda x:datetime.date(x).isocalendar()[1])

df.year.value\_counts().sort\_index(ascending=True).plot(kind='bar')

df.groupby('week\_num')['week\_num'].count().plot()

df['month\_name'] = df['month'].apply(lambda x: calendar.month\_name[x])

# Plotting the bar plot

sns.barplot(x=df['month\_name'].value\_counts().index, y=df['month\_name'].value\_counts().values)

plt.xticks(rotation=45)

plt.show()

colors = sns.color\_palette("husl", len(df['day\_num'].value\_counts()))

# Create the bar plot

sns.barplot(x=df['day\_num'].value\_counts().index, y=df['day\_num'].value\_counts().values, palette=colors)

plt.xticks(rotation=45)

plt.show()

df2=df.copy()

df2.drop(["month\_name","day\_num"],axis=1,inplace=True)

for i, stock\_name in enumerate(df2.columns[1:11], start=1):  # Assuming columns 1 to 5 are the stock prices

    plt.figure(figsize=(8, 6))

    sns.histplot(df2[stock\_name], kde=True, color=f'C{i}')

    plt.title(stock\_name)

    plt.xlabel('Stock Price')

    plt.show()

plt.figure(figsize=(12,6))

def area\_plot():

    for stock\_name in df.columns[1:22]:  # Assuming column 0 is the date

        plt.plot(df['DATE'], df[stock\_name], label=stock\_name)

    plt.xlabel('Year', fontsize=14)

    plt.ylabel('Price', fontsize=14)

    plt.legend()

area\_plot()

plt.show()

n = len(df2.columns)  # Determine the total number of columns in the DataFrame

for name in df2.columns[1:n]:  # Taking first n columns for line plot

    plt.figure(figsize=(10, 5))

    sns.lineplot(x="year", y=name, data=df2)

    plt.title(name)

    plt.ylabel('Close Price')

    plt.show()

# Assuming 'DATE' is the column representing the date in your DataFrame

colors = plt.cm.viridis(np.linspace(0, 1, len(df2.year.unique())))  # Generate colors for each unique year

for name in df2.columns[1:8]:  # Taking first 8 columns for scatter plot

    plt.figure(figsize=(10,5))

    for i, year in enumerate(df2.year.unique()):

        d = df2.loc[df2.year == year, ['DATE', name]]

        plt.scatter(d["DATE"], d[name], color=colors[i], label=year)

    plt.xlabel('Years', fontsize=14)

    plt.ylabel(f'{name} Close Price', fontsize=14)

    plt.legend()

    plt.show()

url = 'https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv'

dataset\_train = pd.read\_csv(url)

training\_set = dataset\_train.iloc[:, 1:2].values

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature\_range=(0,1))

training\_set\_scaled = sc.fit\_transform(training\_set)

X\_train = []

y\_train = []

for i in range(60, 2035):

    X\_train.append(training\_set\_scaled[i-60:i, 0])

    y\_train.append(training\_set\_scaled[i, 0])

X\_train, y\_train = np.array(X\_train), np.array(y\_train)

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

from keras.models import Sequential

from keras.layers import LSTM

from keras.layers import Dropout

from keras.layers import Dense

model = Sequential()

model.add(LSTM(units=50,return\_sequences=True,input\_shape=(X\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(units=50,return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=50,return\_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=50))

model.add(Dropout(0.2))

model.add(Dense(units=1))

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

model.fit(X\_train,y\_train,epochs=100,batch\_size=32)

dataset\_total = pd.concat((dataset\_train['Open'], dataset\_test['Open']), axis = 0)

inputs = dataset\_total[len(dataset\_total) - len(dataset\_test) - 60:].values

inputs = inputs.reshape(-1,1)

inputs = sc.transform(inputs)

X\_test = []

for i in range(60, 76):

    X\_test.append(inputs[i-60:i, 0])

X\_test = np.array(X\_test)

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

predicted\_stock\_price = model.predict(X\_test)

predicted\_stock\_price = sc.inverse\_transform(predicted\_stock\_price)

plt.plot(real\_stock\_price, color = 'black', label = 'TATA Stock Price')

plt.plot(predicted\_stock\_price, color = 'green', label = 'Predicted TATA Stock Price')

plt.title('TATA Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('TATA Stock Price')

plt.legend()

plt.show()