```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data__set = pd.read_csv('/content/Gold Price Prediction Dataset.csv')
data__set.head(5)
data__set.shape
data__set.info()
data__set.fillna(method='ffill', inplace=True)
print(data__set.isnull().sum())

laabel_encoderr = LabelEncoder()
data__set['Date'] = laabel_encoderr.fit_transform(data__set['Date'])

data__set['EU_Trend'] = laabel_encoder.fit_transform(data__set['EU_Trend'])
data__set['OF_Trend'] = laabel_encoderr.fit_transform(data__set['OF_Trend'])
data__set.info()

data__set.drop_duplicates(inplace=True)


print(f"Nmbr of similar rows: {data__set.duplicated().sum()}")
from sklearn.preprocessing import StandardScaler


numerical_columns = ['Open', 'High', 'Low', 'Close', 'Volume', 'SP_open', 'SP_high', 'SP_low',
'SP_close', 'SP_Ajclose','OF_Price',
      'OF_Open', 'OF_High', 'OF_Low', 'OF_Volume', 'EU_Trend_1',
      'OF_Trend_1']


scalerr = StandardScaler()


data__set[numerical_columns] = scaler.fit_transform(data__set[numerical_columns])


print(data__set[numerical_columns].head())
sns.set(font_scale=2)
plt.subplots(figsize=(20,20))
heatt_plott= sns.heatmap(data__set.corr(method='pearson'), annot=True, cmap= 'RdYlGn',
annot_kws={'size': 20})

plt.yticks(fontsize=35)
```

```
plt.xticks(fontsize=35)

plt.show()
correlationss = data__set.corr(method='pearson')
print(correlationss['SP_open'].sort_values(ascending=False).to_string())
columns_dropp = ['EU_Trend_1', 'OF_Trend_1']
existing_columns = data__set.columns

for column in columns_dropp:
    if column in existing_columns:
        data__set = data__set.drop(columns=column)
    else:
        print(f"Column '{column}' not found in DataFrame")

data__set.info()
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split


X = data__set.drop(['Adj Close'], axis=1)
y = data__set['Adj Close']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


linr_regg = LinearRegression()
linr_regg.fit(X_train, y_train)
y_prdt = linr_regg.predict(X_test)
m_s_e = mean_squared_error(y_test, y_prdt)
print("Mean Squared Error:", m_s_e)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

thrsldd = data__set['Adj Close'].mean()
data__set['Adj_Close_Binary'] = (data__set['Adj Close'] > thrsld).astype(int)

X = data__set.drop(['Adj Close', 'Adj_Close_Binary'], axis=1)
y = data__set['Adj_Close_Binary']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

l_regg = LogisticRegression()
l_regg.fit(X_train, y_train)
```

```
y_prdt = l_regg.predict(X_test)
accuracyy = accuracy_score(y_test, y_prdt)
print("Accuracy:", accuracyy)
```