Проект по предметот:

Вовед во наука за податоци

Тема бр.15:

Да се најдат податоците за цената на акциите и ESG score на компанијата Nike за изминатите 4 години. Потоа, да се проучат податоците со користење на tsfresh и истите да се објаснат. Да се направи benchmark на најмалку 3 модели за предикција на цена на акциите на компанијата имајќи ги предвид горенаведените податоци.

Линк до GitHub репозиториум:
https://github.com/andreevskaivana/ITDS-Nike-Stock-Price-Prediction-and-ESG-Score/tree/main
Линк до Youtube видео:
https://www.youtube.com/watch?v=R-bvC_f7dt8

Изработила:

Ивана Андреевска

Бр. на индекс:

203123

Importing and reading the data - Превземање и читање на податоците

За цел на овој проект морав да најдам податоци поврзани со цената на акциите и ESG score на компанијата Nike за изминатите 4 години.Податоците поврзани со акциите на компанијата Nike решив да ги земам од Yahoo Finance кој е доверлив извор,додека пак податоците за ESG Score на компанијата ги земав од https://query2.finance.yahoo.com/v1/finance/esgChart?symbol=NKE .Бидејќи се бара податоците да бидат од последните четири години , истите пред да ги симнам во CSV формат ги исфилтрирав така што временскиот период ќе биде од 14.08.2019 до 14.08.2023.

Откако ги симнаав податоците, морав да ги исчистам истите,а тоа го направив со помош на библиотеката **pandas**.Со командата nike_stock.head() добив идеа од што се содржи мојот dataset и со какви податоци работам.

```
nike_stock=pd.read_csv('/content/drive/MyDrive/VNP_Proekt/NKE.csv')
```

```
nike_stock.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-08-14	81.239998	81.690002	80.510002	81.029999	77.932030	7266100
1	2019-08-15	80.930000	81.300003	79.440002	79.510002	76.470139	6713600
2	2019-08-16	80.089996	80.559998	79.250000	80.279999	77.210701	5649000
3	2019-08-19	82.000000	82.339996	80.830002	81.129997	78.028206	7027900
4	2019-08-20	80.720001	81.230003	79.449997	80.529999	77.451149	5903100

Мојот dataset за цените на акциите ги содржи колоните Date,Open,High,Low,Close,Adj Close и Volume.Објаснувањата за сите овие колони се:

- Date: Специфичен датум за одредена акција.
- Open: Почетната цена на акцијата на тој датум.
- High: Највисоката цена на акцијата на тој датум.
- Low: Најниската цена на акцијата на тој датум.
- Close: Конечната цена на акцијата на тој датум.
- Adj Close: Прилагодената конечна цена, која во предвид зима и остнати фактори(dividends and stock splits).
- Volume: Обемот на тргување,бројот на акции со кои се тргувало на тој датум.

Откако ги исчитав моите податоци поврзани со акциите,морав истото да го направам и за податоците кои се за ESG Score.Бидејќи податоците поврзани со ESG score беа во json

формат прво морав да импоритрам соодветни библиотеки и потоа мојот json фајл да го претворам во pandas dataframe за успешно да работам со моите податоци.Со командата nike_esg.head() добив идеја од што се содржи мојот dataset и со какви податоци работам.

```
nike_esg.head()
```

	PeerGroup	Timestamp	ESGScore	GovernanceScore	EnvironmentScore	SocialScore	SeriesType
0	Textiles & Apparel	1409529600	73.0	75.0	69.0	75.0	symbolSeries
1	Textiles & Apparel	1412121600	73.0	75.0	69.0	75.0	symbolSeries
2	Textiles & Apparel	1414800000	73.0	75.0	69.0	75.0	symbolSeries
3	Textiles & Apparel	1417392000	73.0	75.0	69.0	75.0	symbolSeries
4	Textiles & Apparel	1420070400	73.0	75.0	69.0	75.0	symbolSeries

Data Exploration and Analysis - Анализа и обработка на податоците

После читањето на податоците почнав со нивна обработка и ова вклучуваше форматирање и препроцесирање на податоците,визуелизација и споредба на истите.Прво почнав со работа на податоците поврзани со акциите,со проверка на бројот на колони и редици а потоа со методот **describe** кој врши статистички проверки како наоѓање средна вредност,стандардна девијација,минимална,максимална вредност и квартилите.Ова го направив со цел подобро да се запознаам со колкава количина и какви податоци се работи.Со методот **info** направив проверка за тоа каков тип на објекти е во секоја колона.Следна битна проверка која ја извршив беше за тоа дали има null вредности што за среќа ги немаше.Бидејќи со проверката од info методот дознав дека колоната Date е објект истата морав да ја конвертирам во datetime формат за полесна работа.Откако конвертирав,мојот dataset го сортирав според датум и го ресетирав индексот.

Потоа преминав на визуелизација на податоците.Најпрво ја визуелизирав дистрибуцијата на вредностите кои се наоѓаат во **Close** колоната со помош на хистограм.Друга визуелизација која ја направив беше да претставам како цената на акциите се движела во текот на годините од 2019 до 2023,на x оската е претставен датумот додека на y оската е цената на акциите.

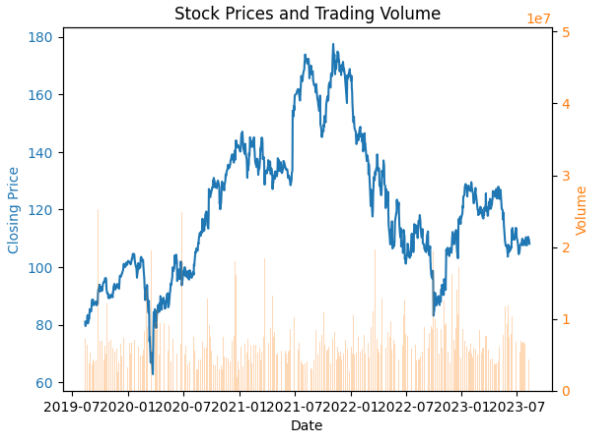
In [101]:
```
plt.plot(nike_stock['Date'], nike_stock['Close'])
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.title('Closing Prices Over Time')
plt.xticks(rotation=45)
plt.show()
```

In [102]:
```
fig, ax1 = plt.subplots()

ax1.plot(nike_stock['Date'], nike_stock['Close'], color='tab:blue')
ax1.set_xlabel('Date')
ax1.set_ylabel('Closing Price', color='tab:blue')
ax1.tick_params(axis='y', labelcolor='tab:blue')

ax2 = ax1.twinx()
ax2.bar(nike_stock['Date'], nike_stock['Volume'], alpha=0.3, color='tab:orange')
ax2.set_ylabel('Volume', color='tab:orange')
ax2.tick_params(axis='y', labelcolor='tab:orange')

plt.title('Stock Prices and Trading Volume')
plt.xticks(rotation=45)
plt.show()
```

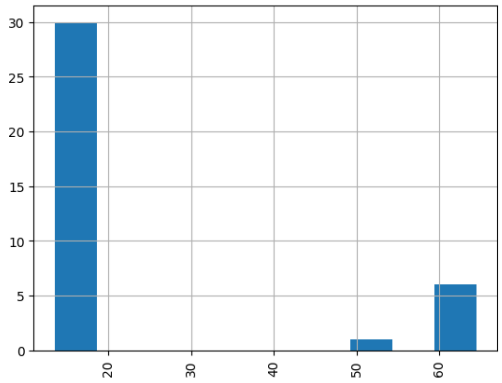


Откако завршив со обработка на податоците поврзани со цените на акциите продолжив со работа на податоците поврзани со ESG Score.Бидејќи форматот на датумот не беше во содветна форма најпрво морав да го конвертирам во обичен datetime формат.Откако го сменив форматот,направив проверка за дали има null вредности, и истите ги решив така што на местата каде има null вредности тоа го заменив со просек на вредностите.Во dataset-от поврзан со ESG Score ги избришав колоните PeerGroup и Series Type бидејќи сметав дека не беа корисни информации и ова ми овозможи полесна работа со

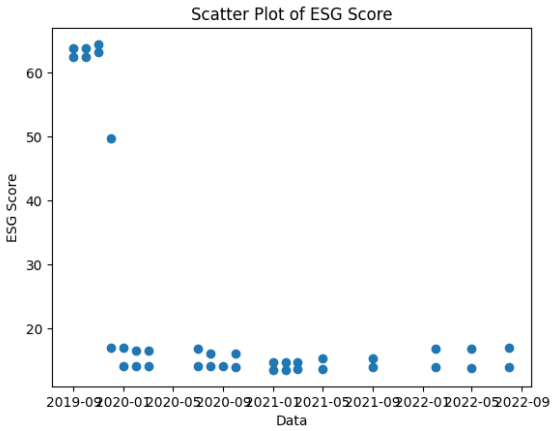
останатите колони-ESG Score, Governance Score, Environment Score и Social Score.Бидејќи овој dataset содржи податоци од 2014,а се бара да се работи со податоци од последните 4 години морав да направам нова филтрација така што ќе се вклучат само вредностите од последните 4 години,ова го направив со поставување на променливи start и end date.За крај на мојата анализа проверив број на редици и колони како и повторно го искористив методот **describe** кој ми даде увид во сумарните статистики:средна вредност,стандардна девијација,минимална,максимална вредност и квартилите.Завршив со визуелизација претставувајќи ги податоците од колоната ESGScore со хистограм,како и со scatter plot.

```
In [117]: #checking distribution
          nike_esg['ESGScore'].hist()
          plt.xticks(rotation=90)
```

```
Out[117]: (array([10., 20., 30., 40., 50., 60., 70.]),
           [Text(10.0, 0, '10'),
            Text(20.0, 0, '20'),
            Text(30.0, 0, '30'),
            Text(40.0, 0, '40'),
            Text(50.0, 0, '50'),
            Text(60.0, 0, '60'),
            Text(70.0, 0, '70')])
```




```
In [118]: plt.scatter(nike_esg['Date'],nike_esg['ESGScore'])
          plt.xlabel('Data')
          plt.ylabel('ESG Score')
          plt.title('Scatter Plot of ESG Score')
          plt.show()
```



Анализа и работа со библиотеката TSFresh

Time series податоци се податоци кои се карактеризираат со секвенцијални набљудувања со текот на времето,и се распространети во различни домени.Извлекувањето значајни информации од time series податоци е од клучно значење за откривање на патерни, правење предвидувања и разбирање на истите. Извлекувањето на карактеристиките на овие податоци е суштински чекор во анализата и вклучува трансформација на необработени податоци во збир на информативни карактеристики што може да се користат во иднина.**TSFRESH (Time Series Feature extraction based on Scalable Hypothesis tests)** е моќна библиотека на Python дизајнирана да го автоматизира процесот на извлекување релевантни карактеристики од податоци за временски серии.TSFresh користи збирка статистички тестови за да одреди кои карактеристики се информативни за дадена временска серија. Овие тестови проценуваат дали одредена шема или однесување е присутно во податоците.

За да можам да работам со **tsfresh** библиотеката најпрво морав да ја инсталирам во мојата работна околина.Откако ја инсталирав tsfresh библиотеката потоа од неа импортирав **extract_features** која овозможува автоматско генерирање на различен број на карактеристики и ја намалува потребата за мануелно извлекување на битните карактеристики.Со извелкувањето на овие карактеристики подобро можеме да ги опишеме нашите податоци, во овој случај временски серијали.Најчесто извелкувањето на овие карактеристики ни дава подобар увид за нашите податоци,како и овозможува полесно понатамошно тренирање на модели со машинско учење кои вршат класификација или регресија на одредени временски серијали.

После импортирањето, морав да предадам карактеристики кои сакам да бидат извлечени,прво за мојот dataset поврзан со акциите.Тоа го направив креирајќи нова променлива **stock_features** каде повикувам **extract features** и ова прима неколку аргументи:

- nike_stock- ова е мојот dataset каде се наоѓаат моите податоци
- column_id =’Date’ - ова служи за да кажеме која од колоните во мојот dataset ќе служи како идентификатор за одредена временска точка (за tsfresh е битно да има уникатен идентификатор за секој податок за да може коректно да ги извлече и организира карактеристиките)
- column_sort=’Date’ - според која колона хронолошки да се сортираат податоците

И за двата аргументи ја одбрав колоната **Date** бидејќи се работи за временски серијал и единствена колона која може да се користи како специфичен идентификатор. Со разгледување на extract_features ги неколку од следните колони кои објаснуваат:

- **Open__variance_larger_than_standard_deviation -** дали варијансата на цените е поголема од стандардната девијација
- **Open__has_duplicate_max -** дали има дупликат максимални вредности во цените
- **Open__has_duplicate_min -** дали има дупликат минимални вредности во цените
- **Open__has_duplicate -** дали има дупликат во цените
- **Open__sum_values -** сума на цените

Анализа и работа со time series - податоци од тип временски серијал

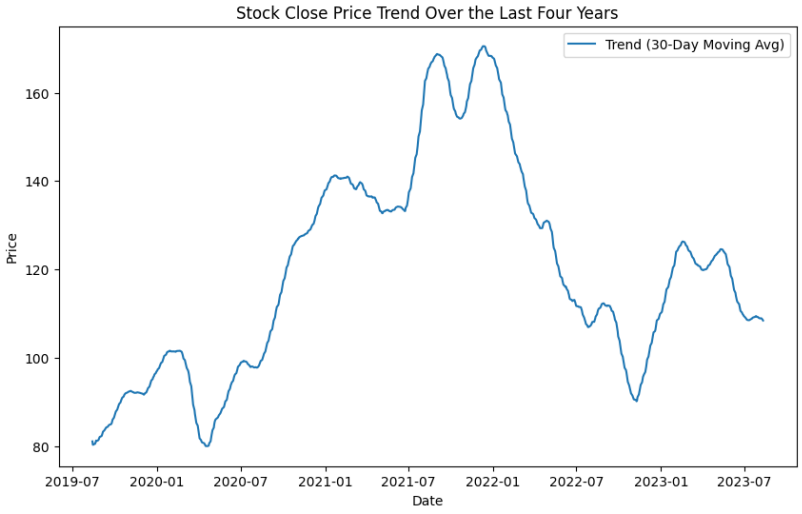
Trend Analysis

Анализата на трендот вклучува гледање на насоката во која се движат цените на акциите со текот на времето. Додека го набљудуваме графикот на кој е претставен тренд, може да забележиме дека цените на акциите генерално се зголемуваат во одредена временска рамка. Постојаното нагорно движење претставува позитивен тренд во перформансите на акциите. Овој тренд може да се должи на фактори како што се зголемената побарувачка за производи на Nike, силните финансиски перформанси или поволните пазарни услови.Како и позитивен тренд,има и негативен каде паѓаат цените на акциите.Анализата на трендот ја поедноставува интерпретацијата на податоците од сложените временски серии.Ова овозможува попрецизни предвидувања, подобро донесување одлуки и подлабоко разбирање на факторите кои го поттикнуваат однесувањето на податоците.

Прво креираме колона Numerical Date која ни ги пресметува бројот на денови кои поминале од најраниот датум во нашиот dataset,ова ни олеснува да знаеме колку време поминало од одреден настан за да калкулираме тренд.Потоа со помош на **moving average** ги елиминираме непотребните податоци т.е “smoothing out fluctuations and noise in the data to reveal underlying trends”, за да можеме да пресметаме moving average ни треба параметар наречен **window size**.Бидејќи работиме со временски серијал,секој ден има различна цена, со помош на moving average ние калкулираме просечна вредност на еден оддел од дефинираните денови.Со дефинирање на window size ние кажуваме колкав број од деновите кои ги имаме заедно во нашиот dataset сакаме да ги групираме заедно.Потоа филтрираме да се вклучат само податоците од последните четири години,каде датумот е поголем или еднаков на пред четири години од максималниот

датум во податоците.На крај повторно правиме плот кој визуелизира како трендот на акциите се однесува во последните четири години пресметан со помош на **moving averages.**Овој plot ни покажува како цената на акциите се променила со текот на времето.Вредностите на x-оската ни покажуваат различни временски периоди од последните четири години и како паѓала/се кревала цената на акцијата тој ден, додека пак на y-оската ни се претставени различните конечни цени на акциите.

```
In [125]: #trend plot over the last four years
plt.figure(figsize=(10, 6))
plt.plot(last_four_years['Date'], last_four_years['Trend'], label=f'Trend ({window_size}-Day Moving Avg)')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Stock Close Price Trend Over the Last Four Years')
plt.legend()
plt.show()
```

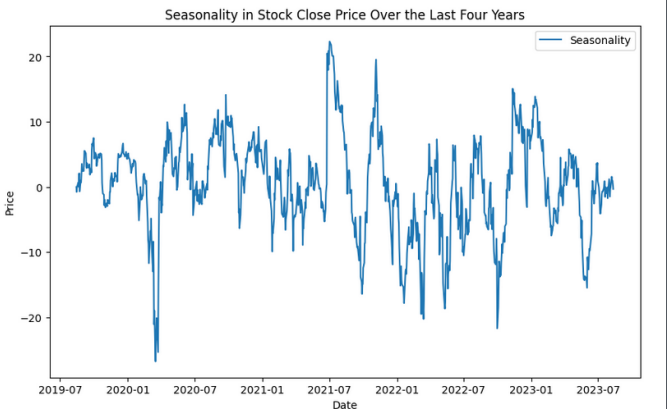


Seasonality Analysis

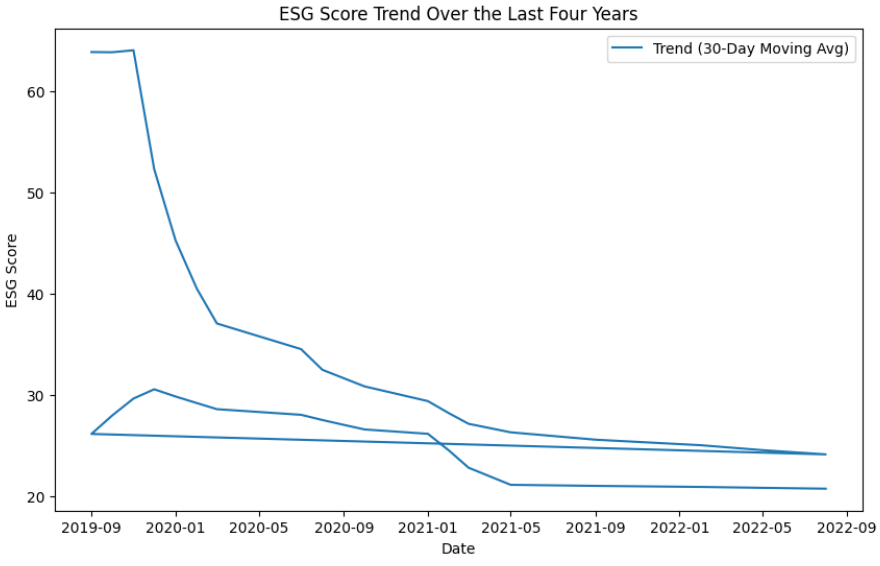
Сезонската анализа во временските серии се однесува на повторувачки и предвидливи настани кои се јавуваат во редовни интервали со текот на времето. Поинаку кажано, тоа е повторлив тренд што се случува редовно, како часовник. Тоа е шема што се повторува на предвидлив начин. Некои акции може да имаат подобри резултати во текот на одредени месеци од годината поради годишни настани, како што се празничните сезони за купување или извештаите за заработката на компанијата. Овие повторливи настани се дел од сезонската анализа на акциите. Трендот го претставува долгорочното движење, додека сезонската состојба ги доловува моделите кои се повторуваат.

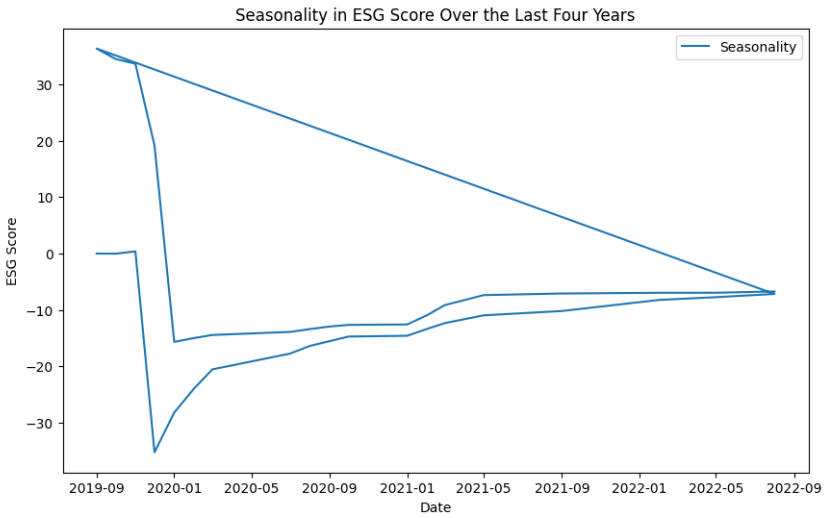
Идејата позади првичната пресметка е тоа што со трендот ги пресметуваме спорите промениве во текот на времето во некој одреден временски интервал.Со одзимање на трендот,ние се фокусираме на она што ни е останато,**краткиот повторлив период во некои одредени интервали.**За крај со помош на плот ја визуелизираме сезоналноста.

```
plt.figure(figsize=(10, 6))
plt.plot(nike_stock['Date'], nike_stock['Seasonality'], label='Seasonality')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Seasonality in Stock Close Price Over the Last Four Years')
plt.legend()
plt.show()
```



Откако завршивме со tsfresh анализа на податоците за цените на акциите истото го правиме и на податоците за ESG Score.Бидејќи и овие податоци се распоредени според датум, истите можеме да ги анализираме како time series податоци односно да пресметаме trend и seasonality за истите.Но најпрво како што направивме и за цените на акциите , така и сега ги извлековме најбитните карактеристики со помош на **extract_features** опцијата od tsfresh.Потоа продолжив со пресметување trend и seasonality и истите резултати ги визуелизирав за како се менувал ESG Score со тек на време во последните 4 години.





Користење различни модели за предикција

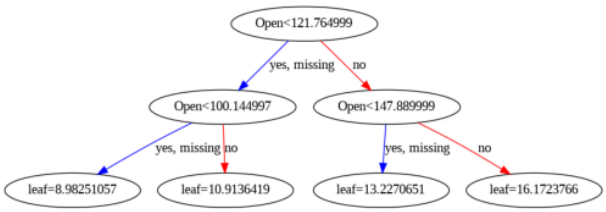
Имајќи ги во предвид горенаведените податоци, следна задача која што ја имав беше да ги предвидам цените на акциите со различни модели.Пред да почнам со предвидување на цените со помош на sns.pairplot ја видов зависноста на колоните од dataset-от за цените на акциите.Како модели за предикција ги одбрав:

- **XGBoost**

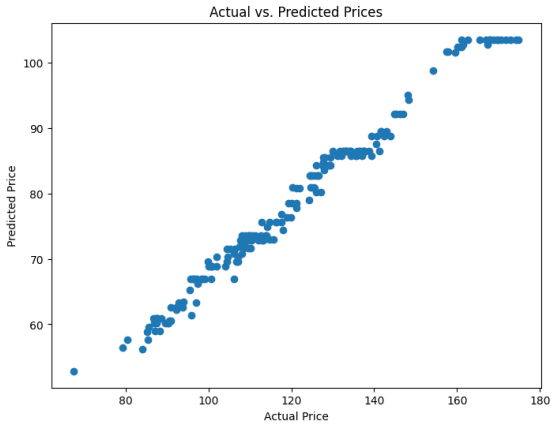
За да може да почнам со работа со XGBoost, морав најпрво да го поделам мојот dataset во трентирачко и тестирачко множество.Колоната на која треба да и се предвиди вредноста е **Close** бидејќи таму се чува кончената цена на акциите.Тренирачкото и тестирачкото множество ги поделив во размер од 80% кои припаѓаат на тестирачко множество , а 20% на тренирачко.Откако го поделив dataset-от следно што направив е го изградив моделот со потребните параметри и истиот го фитував.На крај креирав променлива во која ќе се зачува резултатот од предвидувањата.После градењето на моделот , следна задача ми беше да го евалуирам истиот и да проверам метрики како:mean absolute error,mean squared error и root mean squared error,cross validation. Креирав results data frame кој во себе содржи информации за перформансите на моделот во секоја boosting round како и неговиот RMSE score.Предвидувањето со XGBoost го завршив со неколку визуелизации од кои една од нив е decision tree каде секој јазол во дрвото ни претставува една одлука базирана на некоја карактеристика која ја има нашиот модел.Друга визуелизација е scatter plot која ги споредува цените кои се веќе во dataset-от и предвидените цени.

In [169]:
```
import matplotlib.pyplot as plt
from xgboost import plot_tree

plot_tree(xg_reg, num_trees=0)
plt.rcParams['figure.figsize'] = [1000, 1000]
plt.show()
```



In [171]:
```
plt.figure(figsize=(8, 6))
plt.scatter(Y_test, y_pred)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs. Predicted Prices')
plt.show()
```



- **LGBM**

Како и со XGBoost , така и со овој модел морав прво да започнам со делење на множеството на тестирачко и тренирачко,повторно колоната која се предвидува е Close,а тренирачкото множество е 80% ,а тестирачкото е 20%.Со помош на SimpleImputer во редиците каде што нема вредности,ги пополнив со нивната средна вредност.Со импортирање на LGBMRegressor го изградив и фитував моделот.За крај направив евалуација на моделот и ги добив пресметките од mean absolute error,mean squared error.root mean squared error.

- **LSTM**

Повторно почнав со поделба на тренирачко и тестирачко множество како до сега , па преминав на градењењ на моделот.За да можам да го изградам моделот од библиотеката **keras.models** морав да ги импортирам Sequential,Dense и LSTM за потоа да можам истиот модел соодветно да го компајлирам.Моделот го фитував и ги внесов

потребните параметри како на пример 30 епохи,Завршив со евалуација и визуелизација на моделот.

In [189]:
```
plt.figure(figsize=(10, 6))
y_pred = model.predict(X_test)
plt.plot(y_pred, label='predict')
plt.plot(Y_test, label='true')
plt.legend()
plt.show()
```

```
7/7 [==============================] - 0s 5ms/step
```