IBM
AI ANALYST
USE CASES – CIA I

1. How to retrieve stock price from google finance using panda's data reader. Do the analysis of stock in the format of plotting its high, low, close, volume values in table and a chart.

In [2]:
```
import pandas_datareader as pdr
import pandas as pd
import matplotlib.pyplot as plt
import datetime
```

In [14]:
```
starting_date = '2019/01/01'
ending_date = '2021/12/31'
symbol = 'BMW.DE'
data_source = 'yahoo'
bmw = pdr.data.DataReader(symbol,data_source,starting_date,ending_date)
bmw
```

Out[14]:

Date	High	Low	Open	Close	Volume	Adj Close
2019-01-02	70.660004	68.790001	70.629997	69.739998	1429230.0	55.668568
2019-01-03	69.809998	69.019997	69.220001	69.050003	1426463.0	55.117790
2019-01-04	71.769997	69.639999	69.800003	71.709999	1857639.0	57.241074
2019-01-07	72.320000	71.269997	71.750000	72.120003	1238553.0	57.568359
2019-01-08	73.680000	71.279999	71.849998	72.209999	1865750.0	57.640186
...	...	...	...	...	...	...
2021-12-23	89.459999	87.980003	88.000000	89.169998	823550.0	82.882790
2021-12-27	90.110001	88.610001	88.720001	90.000000	396304.0	83.654266
2021-12-28	90.690002	89.750000	90.089996	89.949997	442096.0	83.607788
2021-12-29	89.970001	88.870003	89.889999	89.199997	419820.0	82.910675
2021-12-30	89.500000	88.120003	89.389999	88.489998	598323.0	82.250732

760 rows × 6 columns

In [15]:
```
bmw.head()
```

Out[15]:

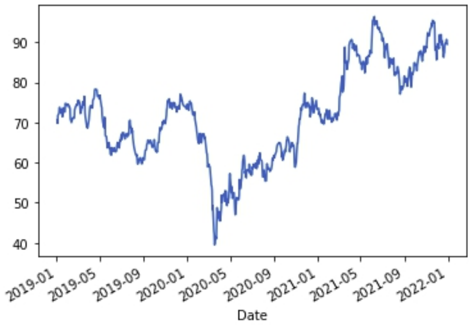
Date	High	Low	Open	Close	Volume	Adj Close
2019-01-02	70.660004	68.790001	70.629997	69.739998	1429230.0	55.668568
2019-01-03	69.809998	69.019997	69.220001	69.050003	1426463.0	55.117790
2019-01-04	71.769997	69.639999	69.800003	71.709999	1857639.0	57.241074
2019-01-07	72.320000	71.269997	71.750000	72.120003	1238553.0	57.568359
2019-01-08	73.680000	71.279999	71.849998	72.209999	1865750.0	57.640186

In [16]:
```
bmw.tail()
```

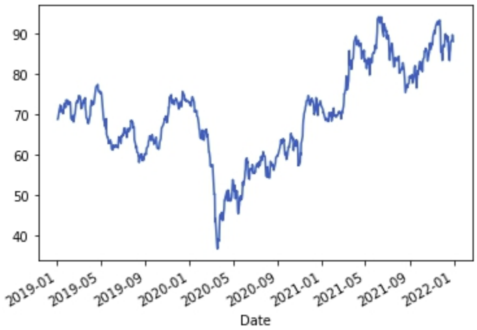
Out[16]:

Date	High	Low	Open	Close	Volume	Adj Close
2021-12-23	89.459999	87.980003	88.000000	89.169998	823550.0	82.882790
2021-12-27	90.110001	88.610001	88.720001	90.000000	396304.0	83.654266
2021-12-28	90.690002	89.750000	90.089996	89.949997	442096.0	83.607788
2021-12-29	89.970001	88.870003	89.889999	89.199997	419820.0	82.910675
2021-12-30	89.500000	88.120003	89.389999	88.489998	598323.0	82.250732

In [23]:
```
bmw['High'].plot()
plt.show()
```



In [24]:
```
bmw['Low'].plot()
plt.show()
```



In [21]:
```
bmw['Open'].plot(label='Stock Prices at Open')
bmw['High'].plot(label='Stock Prices on High')
bmw['Low'].plot(label='Stock Prices on Low')
bmw['Close'].plot(label='Stocks Prices At Closing',figsize=(16,5))
plt.legend()
plt.title('BMW stock prices')
plt.ylabel('Stock Prices')
plt.show()
```

2. We have a dataset containing prices of used BMW cars. We are going to analyze this dataset and build a prediction function that can predict a price by taking mileage and age of the car as input. We will use sklearn train_test_split method to split training and testing dataset

In [28]:
```
import pandas as pd
```

In [29]:
```
df = pd.read_csv(r'Downloads/bmw.csv')
df
```

Out[29]:

	Model	Year	Price($)	Mileage
0	5 Series	2014	11200	67068
1	6 Series	2018	27000	14827
2	5 Series	2016	16000	62794
3	1 Series	2017	12750	26676
4	7 Series	2014	14500	39554
...	...	...	...	...
10776	X3	2016	19000	40818
10777	5 Series	2016	14600	42947
10778	3 Series	2017	13100	25468
10779	1 Series	2014	9930	45000
10780	X1	2017	15981	59432

10781 rows × 4 columns

In [30]:
```
df.head()
```

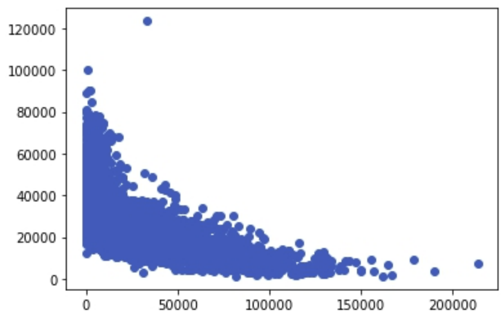
Out[30]:

	Model	Year	Price($)	Mileage
0	5 Series	2014	11200	67068
1	6 Series	2018	27000	14827
2	5 Series	2016	16000	62794
3	1 Series	2017	12750	26676
4	7 Series	2014	14500	39554

In [31]:
```
import matplotlib.pyplot as plt
```

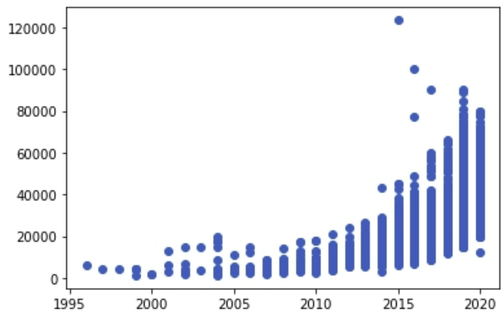
In [32]:
```
plt.scatter(df['Mileage'],df['Price($)'])
```

Out[32]: <matplotlib.collections.PathCollection at 0x7fb891fd3400>



In [34]:
```
plt.scatter(df['Year'],df['Price($)'])
```

Out[34]: <matplotlib.collections.PathCollection at 0x7fb891e4ed40>



In [35]:
```
X = df[['Mileage','Year']]
```

In [36]:
```
y = df['Price($)']
```

In [37]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

In [38]:
```
X_train
```

Out[38]:

	Mileage	Year
5234	4500	2020
10556	49373	2017
9782	18000	2018
7678	88000	2013
10463	56232	2014
...	...	...
4270	2630	2019
4120	45919	2016
9439	23407	2016
9811	59000	2014
8680	20905	2016

7546 rows × 2 columns

In [39]:
```
X_test
```

Out[39]:

	Mileage	Year
9125	28772	2017
6263	10	2020
1459	123	2019
3031	23000	2019
6564	43626	2015
...	...	...
1191	37500	2016
6260	11812	2019
7253	14616	2018
3175	11449	2019
2105	27225	2017

3235 rows × 2 columns

In [40]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[40]:
```
▾ LinearRegression
LinearRegression()
```

In [41]:
```
model.predict(X_test)
```

Out[41]:
```
array([22045.54641097, 31616.6013637 , 29720.37281587, ...,
       25859.94377313, 28173.32588921, 22256.85504558])
```

In [43]:
```
model.score(X_test, y_test)
```

Out[43]:
```
0.42070202397908885
```

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=10)
X_test
```

Out[44]:

	Mileage	Year
5128	5500	2020
4976	34697	2017
4856	37000	2016
2479	11	2020
4304	4155	2019
...	...	...
433	14052	2018
392	48697	2016
2785	11	2019
4033	105	2019
3374	36779	2017

3235 rows × 2 columns

3. A bank manager is given a data set containing records of 1000s of applicants who have applied for a loan. How can AI help the manager understand which loans he can approve? Explain.

This problem statement can be solved using the KNN algorithm, which will classify the applicant's loan request into two classes:

1. Approved
2. Disapproved

K Nearest Neighbour is a Supervised Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.

The following steps can be carried out to predict whether a loan must be approved or not:

Data Extraction: At this stage data is either collected through a survey or web scraping is performed. Data about the customers must be collected. This includes their account balance, credit amount, age, occupation, loan records, etc. By using this data, we can predict whether or not to approve the loan of an applicant.

Data Cleaning: At this stage, the redundant variables must be removed. Some of these variables are not essential in predicting the loan of an applicant, for example, variables such as Telephone, Concurrent credits, etc. Such variables must be removed because they will only increase the complexity of the Machine Learning model.

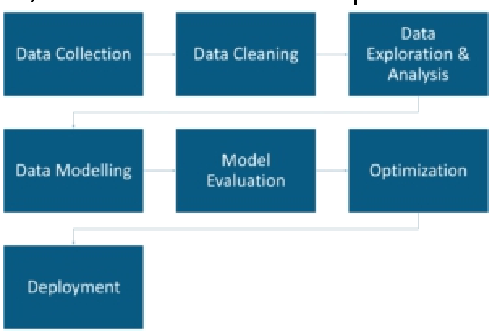
Data Exploration & Analysis: This is the most important step in AI. Here you study the relationship between various predictor variables. For example, if a person has a history of unpaid loans, then the chances are that he might not get approval on his loan applicant. Such patterns must be detected and understood at this stage.

Building a Machine Learning model: There are n number of machine learning algorithms that can be used for predicting whether an applicant loan request is approved or not. One such example is the K-Nearest Neighbor, which is a classification and a regression algorithm. It will classify the applicant's loan request into two classes, namely, Approved and Disapproved.

Model Evaluation: Here, you basically test the efficiency of the machine learning model. If there is any room for improvement, then parameter tuning is performed. This improves the accuracy of the model.

4. You've won a 2-million-dollar worth lottery' we all get such spam messages. How can AI be used to detect and filter out such spam messages?

To understand spam detection, let's take the example of Gmail.



A machine learning process always begins with data collection. We all know the data Google has, is not obviously in paper files. They have data centers which maintain the customer's data. Data such as email content, header, sender, etc are stored.

This is followed by data cleaning. It is essential to get rid of unnecessary stop words and punctuations so that only the relevant data is used for creating a precise machine learning model. Therefore, in this stage stop words such as 'the', 'and', 'a' are removed. The text is formatted in such a way that it can be analyzed.

After data cleaning comes data exploration and analysis. Many a time, certain words or phrases are frequently used in spam emails. Words like "lottery", "earn", "full-refund" indicate that the email is more likely to be a spam one. Such words and co-relations must be understood in this stage.

After retrieving useful insights from data, a machine learning model is built. For classifying emails as either spam or non-spam you can use machine learning algorithms like Logistic Regression, Naïve Bayes, etc. The machine learning model is built using the training dataset. This data is used to train the model and make it learn by using past user email data.

This stage is followed by model evaluation. In this phase, the model is tested using the testing data set, which is nothing but a new set of emails. After which the machine learning model is graded based on the accuracy with which it was able to classify the emails correctly.

Once the evaluation is over, any further improvement in the model can be achieved by tuning a few variables/parameters. This stage is also known as parameter tuning. Here, you basically try to improve the efficiency of the machine learning model by tweaking a few parameters that you used to build the model.

The last stage is deployment. Here the model is deployed to the end users, where it processes emails in real time and predicts whether the email is spam or non-spam.

5. 'Customers who bought this also bought this…' we often see this when we shop on Amazon. What is the logic behind recommendation engines?

What is a Recommendation Engine:

A product recommendation engine is essentially a solution that allows marketers to offer their customers relevant product recommendations in real-time. As powerful data filtering tools, recommendation systems use algorithms and data analysis techniques to recommend the most relevant product/items to a particular user.

The main aim of any recommendation engine is to stimulate demand and actively engage users. Primarily a component of an eCommerce personalization strategy, recommendation engines dynamically populate various products onto websites, apps, or emails, thus enhancing the customer experience. These kinds of varied and omnichannel recommendations are made based on multiple data points such as customer preferences, past transaction history, attributes, or situational context.

Recommender systems can be used across multiple verticals such as e-commerce, entertainment, mobile apps, education, and more (*discussed in detail later*). In general, a recommendation engine can be helpful in any situation where there is a need to give users personalized suggestions and advice.

How does a Recommendation Engine Work:

One of the crucial components behind the working of a product recommendation engine is the recommender function, which considers specific information about the user and predicts the rating that the user might assign to a product.

Having the ability to predict user ratings, even before the user has provided one, makes recommender systems a powerful tool.

It uses specialized algorithms and techniques that can support even the largest of product catalogs. Driven by an orchestration layer, the recommendation engine can intelligently select which filters and algorithms to apply in any given situation for a specific customer. It allows marketers to maximize conversions and also their average order value.

Typically, a recommendation engine processes data through the below four phases-

- Collection : Data collected here can be either explicit such as data fed by users (ratings and comments on products) or implicit such as page views, order history/return history, and cart events.
- Storing : The type of data you use to create recommendations can help you decide the kind of storage you should use, like the NoSQL database, a standard SQL database, or object storage.
- Analyzing : The recommender system analyzes and finds items with similar user engagement data by filtering it using different analysis methods such as batch analysis, real-time analysis, or near-real-time system analysis.
- Filtering : The last step is to filter the data to get the relevant information required to provide recommendations to the user. And for enabling this, you will need to choose an algorithm suiting the recommendation engine from the list of algorithms explained in the next section.

Types Of Recommender Systems

There are many problems solved by machine learning, but making product recommendations is a widely recognized application of machine learning. There are mainly three essential types of recommendation engines –

1. Collaborative Filtering

The collaborative filtering method is based on collecting and analyzing information based on behaviors, activities, or user preferences and predicting what they will like based on the similarity with other users. The prediction is done using various predictive maintenance machine learning techniques.

The two types of collaborative filtering techniques are –

i. User-User collaborative filtering

ii. Item-Item collaborative filtering

One of the main advantages of the collaborative filtering approach is that it can recommend complex items accurately, such as movies, without requiring an *understanding* of the item itself as it does not depend on machine analyzable content.

2. Content-Based Filtering

Content-based filtering methods are mainly based on the description of an item and a profile of the user’s preferred choices. In content-based filtering, keywords are used to describe the items, whereas a user profile is built to state the type of item this user likes.

For example, if a user likes to watch movies such as *Mission Impossible,* then the recommender system recommends movies of the *action* genre or movies of *Tom Cruise.*

The critical premise of content-based filtering is that if you like an item, you will also like a *similar* item. This approach has its roots mainly in information retrieval and information filtering research.

3. Hybrid Recommendation Systems

Hybrid Recommendation engines are essentially the combination of diverse rating and sorting algorithms. For instance, a hybrid recommendation engine could use collaborative filtering and product-based filtering in tandem to recommend a broader range of products to customers with accurate precision.

Netflix is an excellent example of a hybrid recommendation system as they make recommendations by:

- Comparing the watching and searching habits of users and finding similar users on that platform, thus making use of collaborative filtering
- Recommending such shows/movies which share common characteristics with the ones rated highly by the user. It is how they make use of content-based filtering.

Compared to pure collaborative and content-based methods, hybrid methods can provide more accurate recommendations. They can also overcome the common issues in recommendation systems such as cold start and the data paucity troubles.