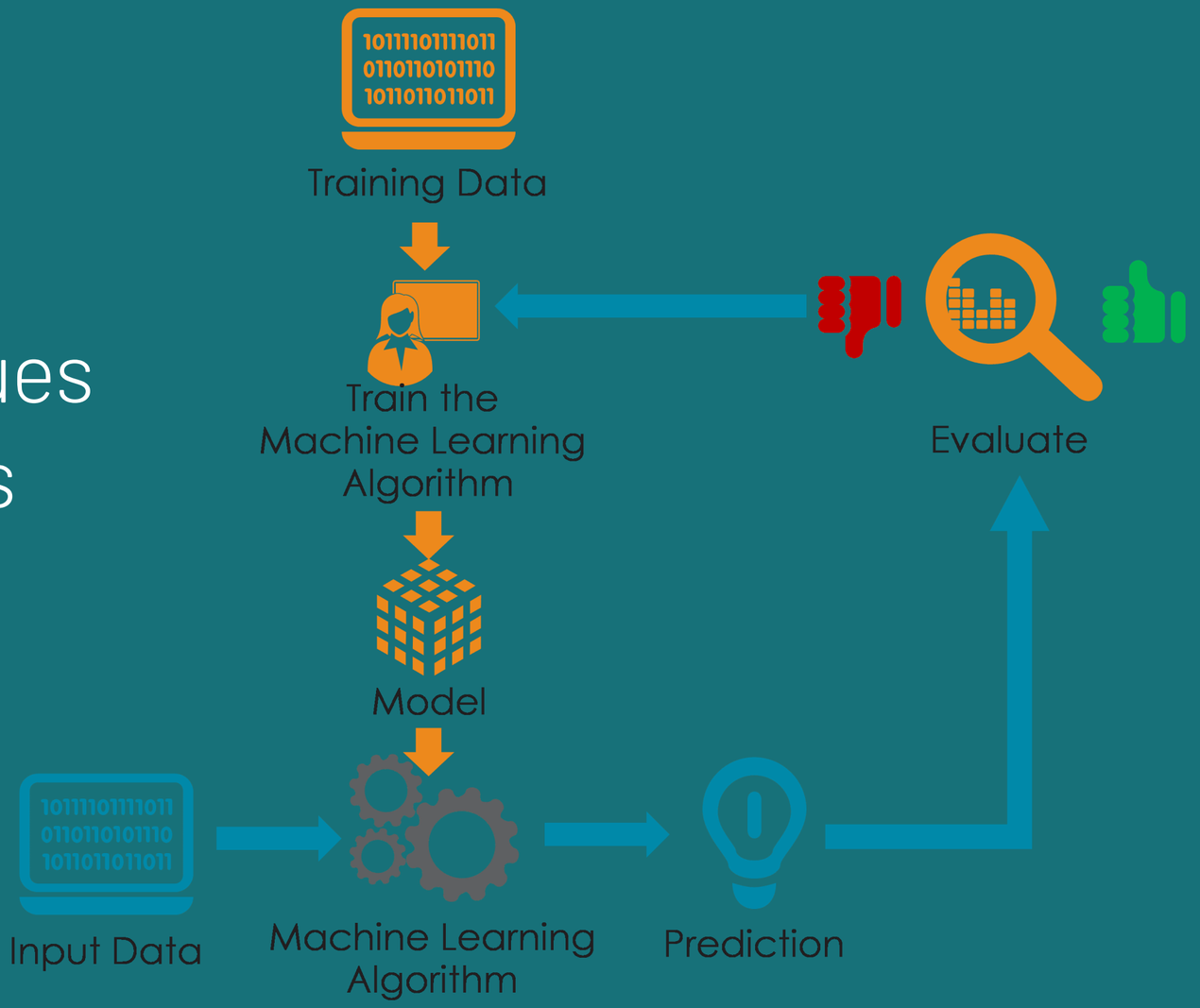
WILL MY TRAIN BE LATE?

PRESENTED BY:
DANIEL ASHOROV
&
ORI GOLDENBERG

TimeLine:

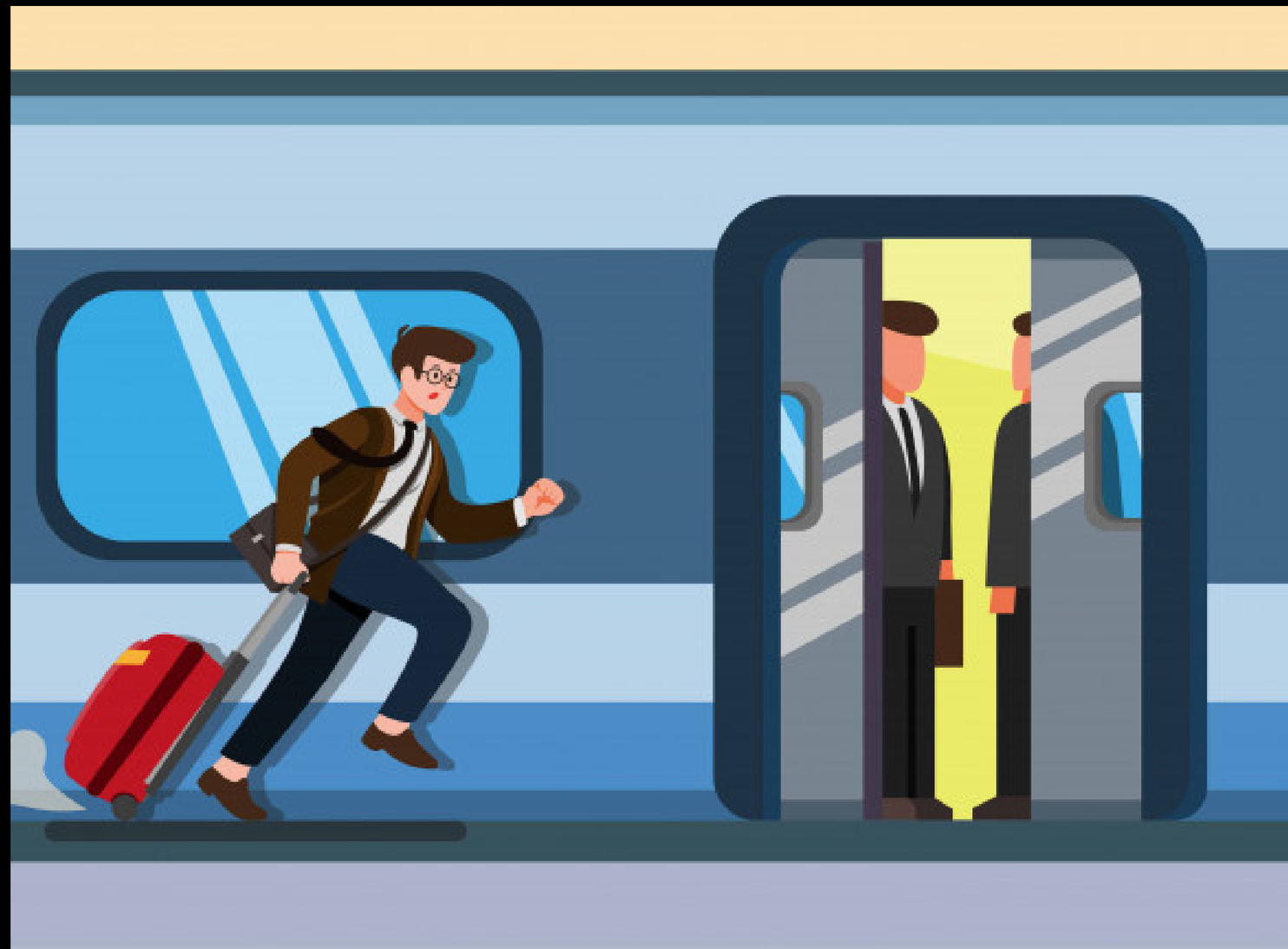
1. Find data resources
2. Clean data from NaN values
3. Exploratory Data Analysis
4. Machine Learning
5. Conclusions



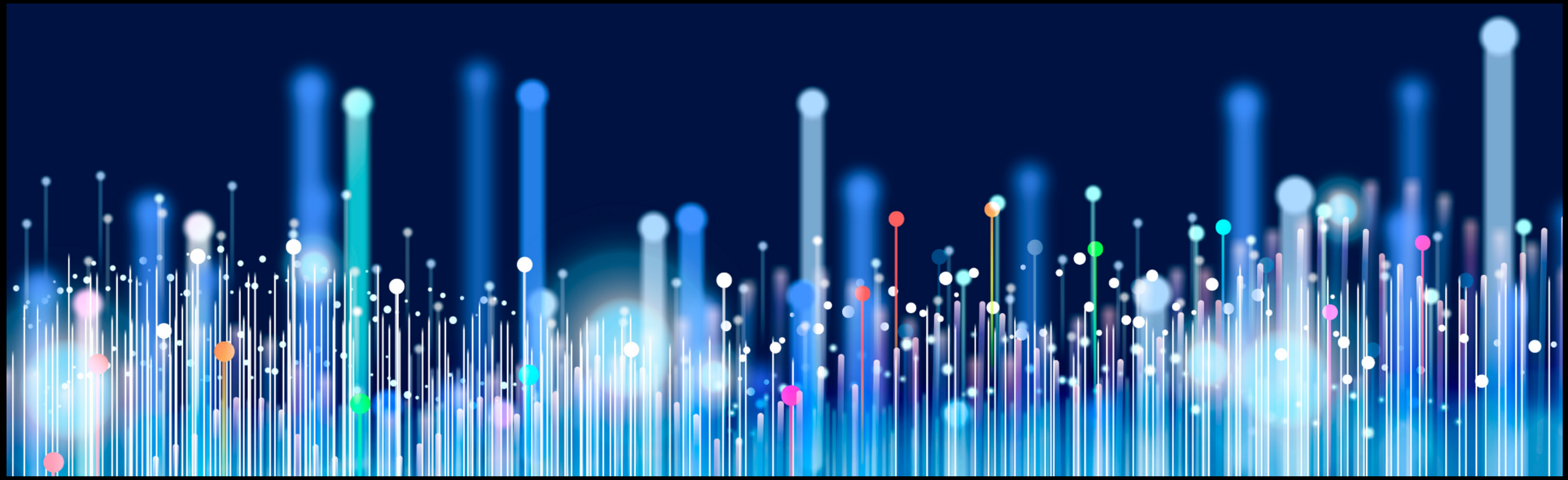
Presenting the prediction question

1. We decided to find answer to one of the biggest problems in Israel nowadays: will the train I need to get on, will be late?

2. In our project we will introduce the answer and predict the probability for you to leave the station at time



Data sources:



https://data.gov.il/dataset/

At the beginning of the project we tried to scrap some information out of the internet but realized that there is no information published.after a few days, looking for a solution we found that the Israeli government allows data scientists to access their data stocks. after a few hours of search we found "Israel railways" company csv started in 2015 until 2021. (because of the israeli government policy we weren't able to access to any train schedule). the CSV includes every train station in Israel and show the schedule for each month every year.

CSV file upload

So now that we have the right CSV, we would like to check if it matches our terms. we used pandas library as pd to see its dataframe.

Reading Dataset

In [4]:
```
df=pd.read_csv("Data_Updated.csv")
df.head()
```

Out[4]:

	_id	Year	Month	Station_Name	Leaving_Status	Num_of_Traffic
0	1905.0	2016.0	1.0	Ofakim	Late	6.0
1	1906.0	2016.0	1.0	Ofakim	In-Time	38.0
2	1907.0	2016.0	1.0	Ashdod_Ad_Halom	Late	58.0
3	1908.0	2016.0	1.0	Ashdod_Ad_Halom	In-Time	1744.0
4	1909.0	2016.0	1.0	Ashdod_Ad_Halom	Precede	135.0

Data Cleaning

After we uploaded the data file,we can see that the "id" column is not necessary (we already have numerical order), so we used "df.drop" command to drop out this column

```
[5]: # Dropping the id column
     df.drop("_id", axis=1, inplace=True)

[6]: df.shape

Out[6]: (10928, 5)

[7]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10928 entries, 0 to 10927
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Year             10920 non-null  float64
 1   Month            10920 non-null  float64
 2   Station_Name     10920 non-null  object
 3   Leaving_Status   10920 non-null  object
 4   Num_of_Traffic   10920 non-null  float64
dtypes: float64(3), object(2)
memory usage: 427.0+ KB
```

As we can see, we have in the cells only numeric values. data cells with missing values would be presented as "NaN", we want to delete them from our CSV. with the "df.dropna" command.

```
[9]: df[df.Year.isnull()]

Out[9]:
```

	Year	Month	Station_Name	Leaving_Status	Num_of_Traffic
10920	NaN	NaN	NaN	NaN	NaN
10921	NaN	NaN	NaN	NaN	NaN
10922	NaN	NaN	NaN	NaN	NaN
10923	NaN	NaN	NaN	NaN	NaN
10924	NaN	NaN	NaN	NaN	NaN
10925	NaN	NaN	NaN	NaN	NaN
10926	NaN	NaN	NaN	NaN	NaN
10927	NaN	NaN	NaN	NaN	NaN

We can see that there are 8 records that are completely **NULL**. So we have to

```
10]: # Dropping Null records
     df.dropna(inplace=True)
```

After removing the missing cells, we want to check if we have duplicated values.we used the "df.duplicated" command for that. luckily, no duplicates found!

Duplicate Rows

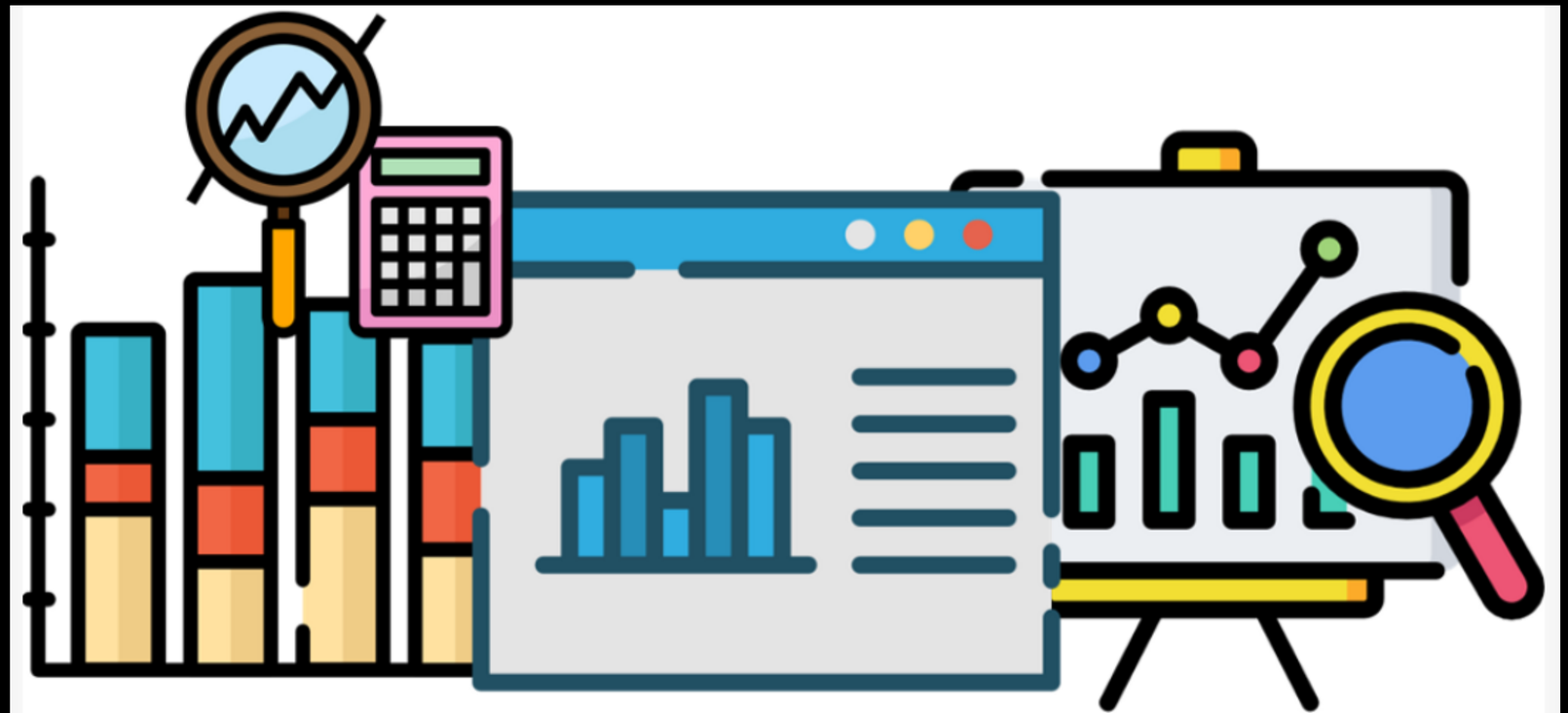
```
df[df.duplicated()]

1]:
```

Year	Month	Station_Name	Leaving_Stat

Exploratory Data Analysis

In this part we will introduce our graphs and their consequences.



About our dataset

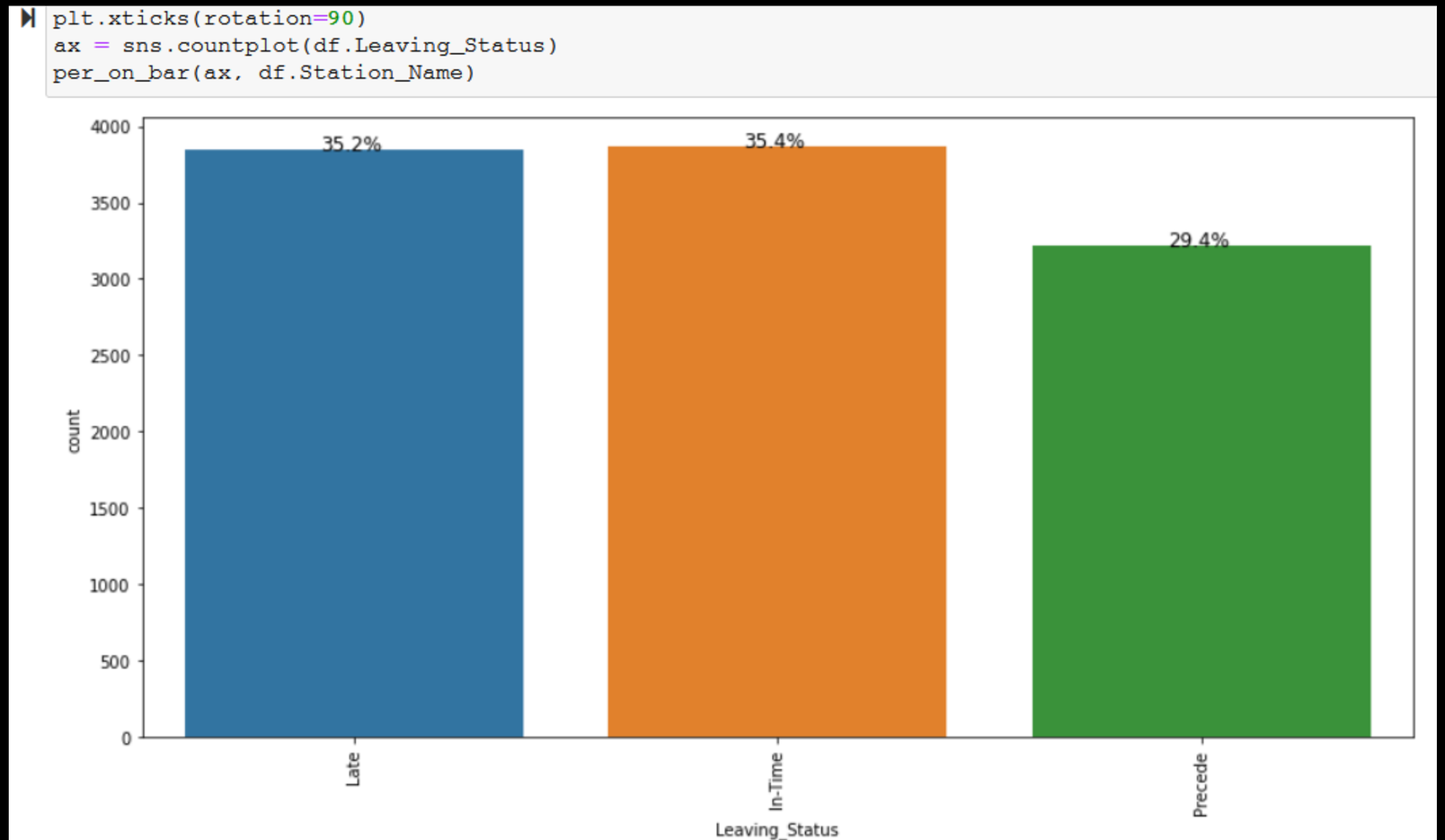
Year	Month	Station_Name	Leaving_Status	Num_of_Traffic

1. year - the year of our train status
2. month - the month in a chosen year and the train station status
3. station name - name of train station
4. leaving status - what is the status of leaving train in my station
5. num of traffic - amount of traffic status in each station

General stations status (Visualisation)

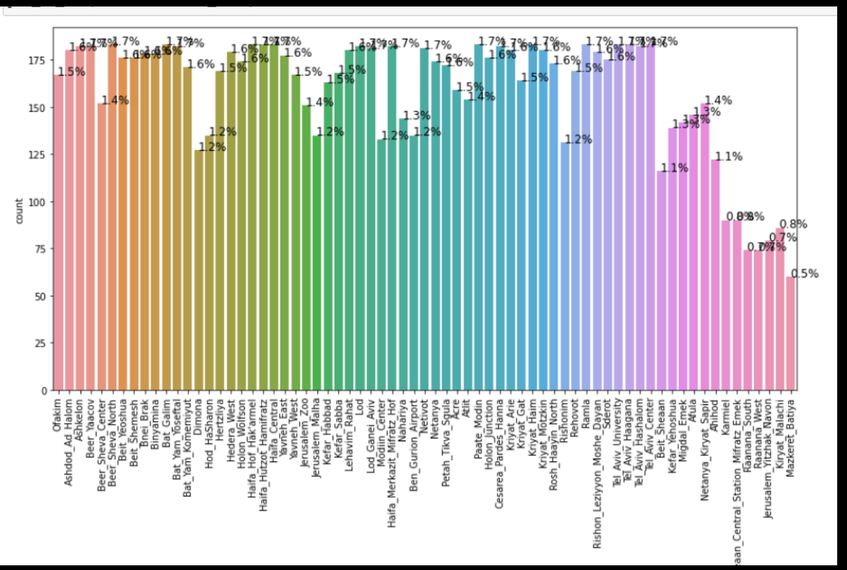
we decided in the first step to present our whole train stations status to see how many trains were late leaving the station and how many train left by-time.

```
plt.xticks(rotation=90)
ax = sns.countplot(df.Leaving_Status)
per_on_bar(ax, df.Station_Name)
```



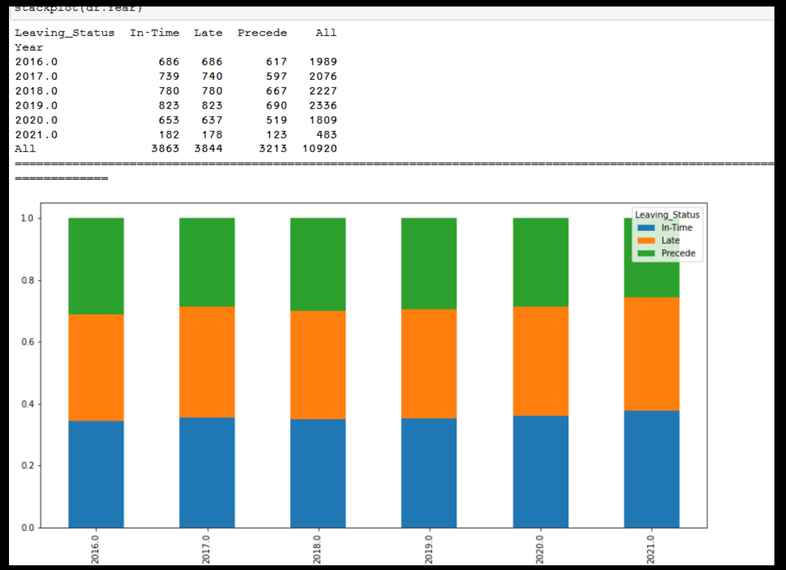
Graphs - what is my status segment?

In the first graph , we choose xticks graph to show by precent what is the part of each station in our graph in % .(whole stations are 100%). for the second and third graphs we choose countplot to make us be able to count numbers of time leaving status



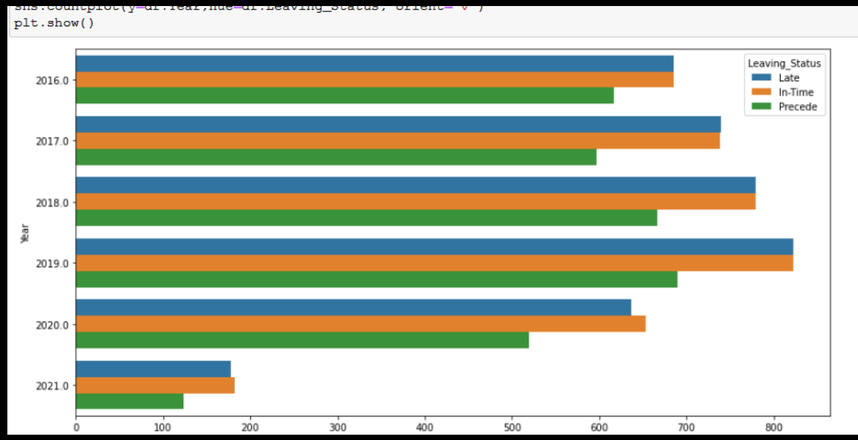
Graph 1

Introduce how many statuses collected in each station.



Graph 2

Introduce yearly percentage graph of general status in the whole stations.

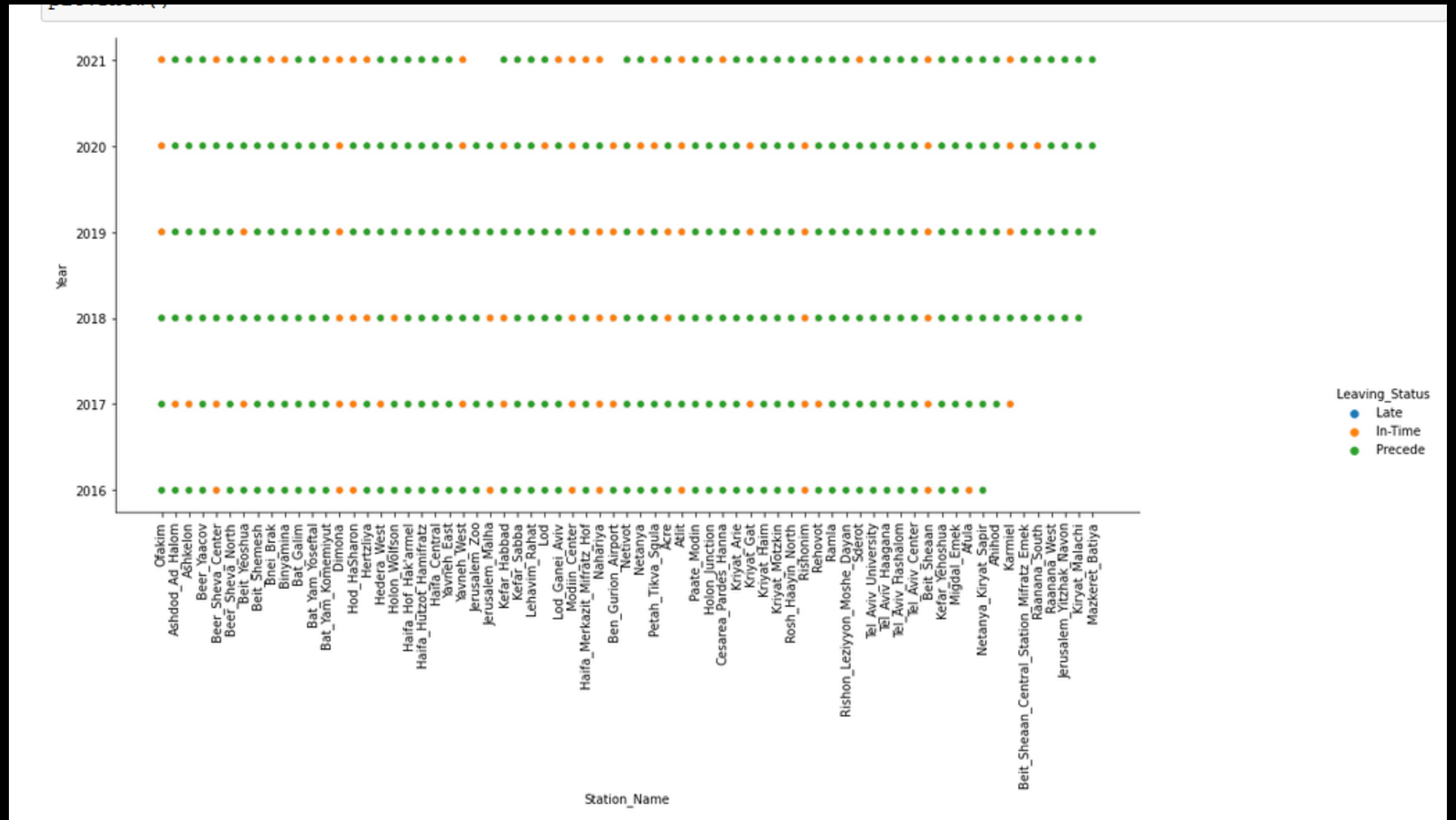


Graph 3

show each year what was the extact number of status generally.

First conclusion - what was the worst year?

As we have seen in the last 3 graphs, and presented in the last summary graph, the 2019 year was the worst your for the train customers. we can also see that this period began in 2017. almost 2 years of trains that leave the station after the original leaving time and caused damage to the customers.

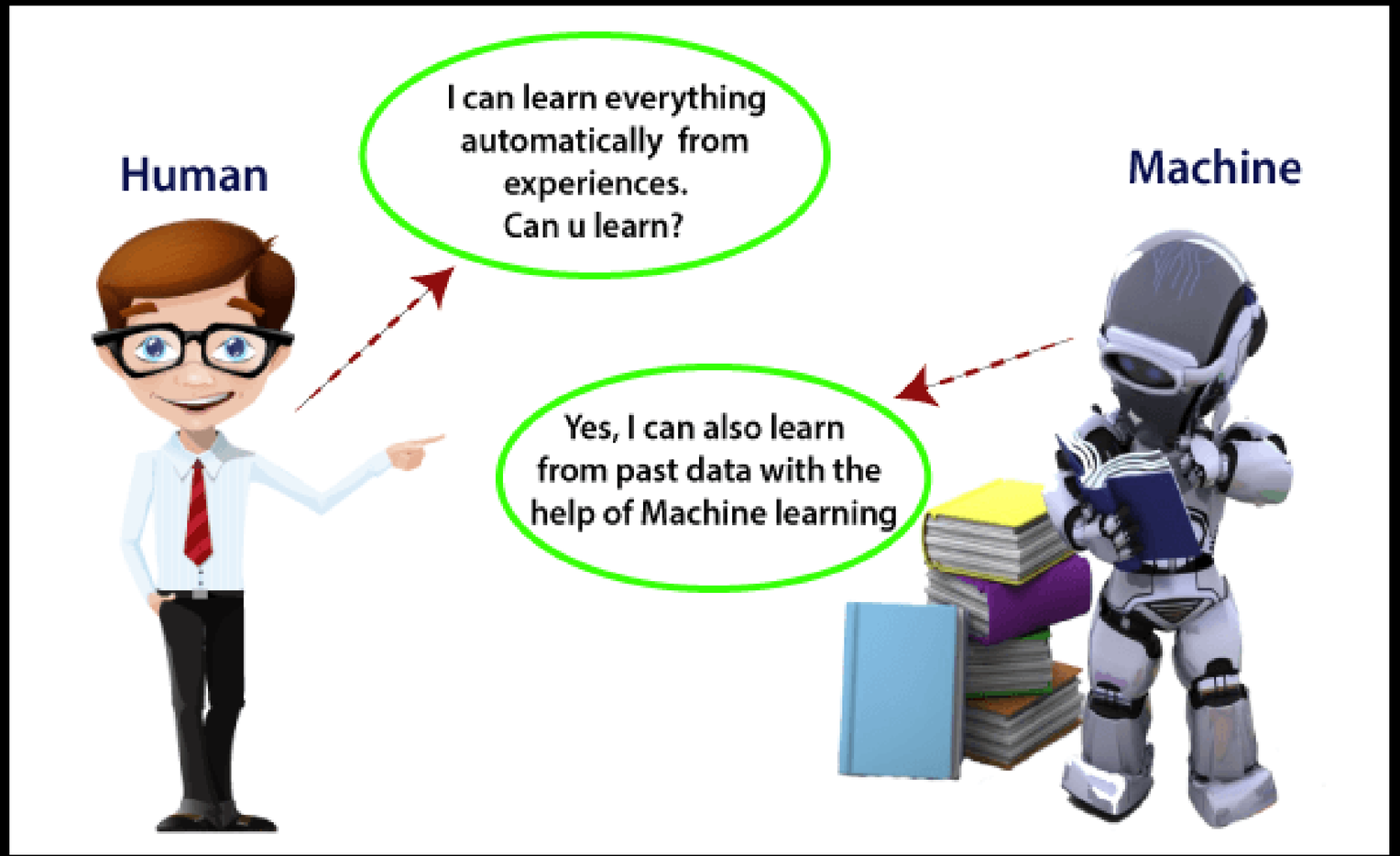


Machine Learning

So after we used math graphs to show information about the status of late leaving trains, we would like to use some machine learning algorithms to predict late train stations in the future....

Machine Learning

Machine learning is a technique that "train" the code by using information from the past to predict by statistics and probability the answer we would like to have.



Label Encoding

inorder to train our model to predict late leaving, we should convert the leaving status of each station to numeric value the the machine would be able to work with. we set : leave in time as 0 , late as 1 , precede as 2 .(we used label encoder to convert the values).

```
# Converting textual data to inetgers
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

categorical_col = ["Station_Name", "Leaving_Status"]

for i in categorical_col:
    df[i] = encoder.fit_transform(df[i])

df.head()
```

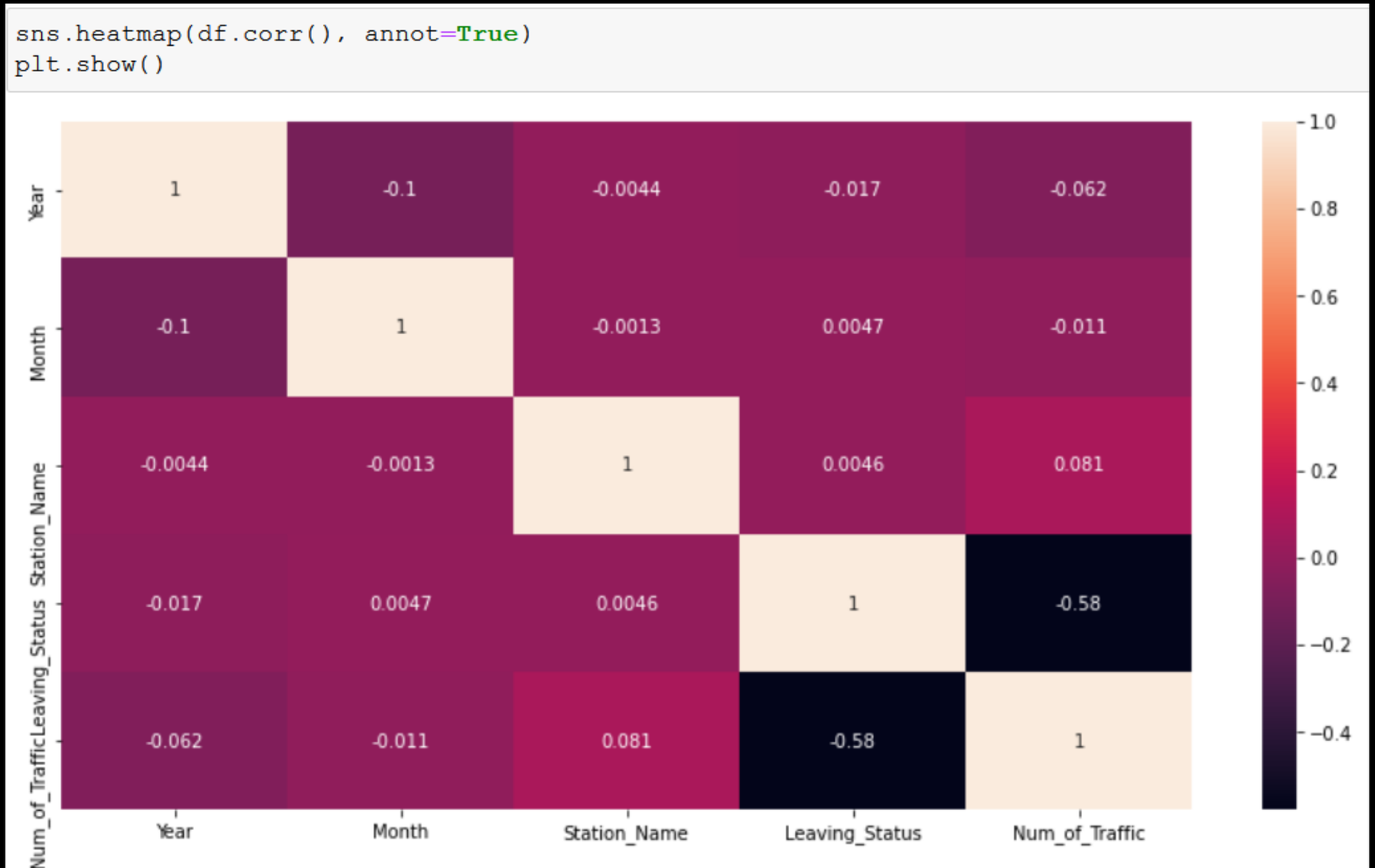
]:

	Year	Month	Station_Name	Leaving_Status	Num_of_Traffic
0	2016.0	1.0	52	1	6.0
1	2016.0	1.0	52	0	38.0
2	2016.0	1.0	3	1	58.0
3	2016.0	1.0	3	0	1744.0
4	2016.0	1.0	3	2	135.0

Feature Selection

what features shall we choose to get the extact data? we will find them by heatmap!

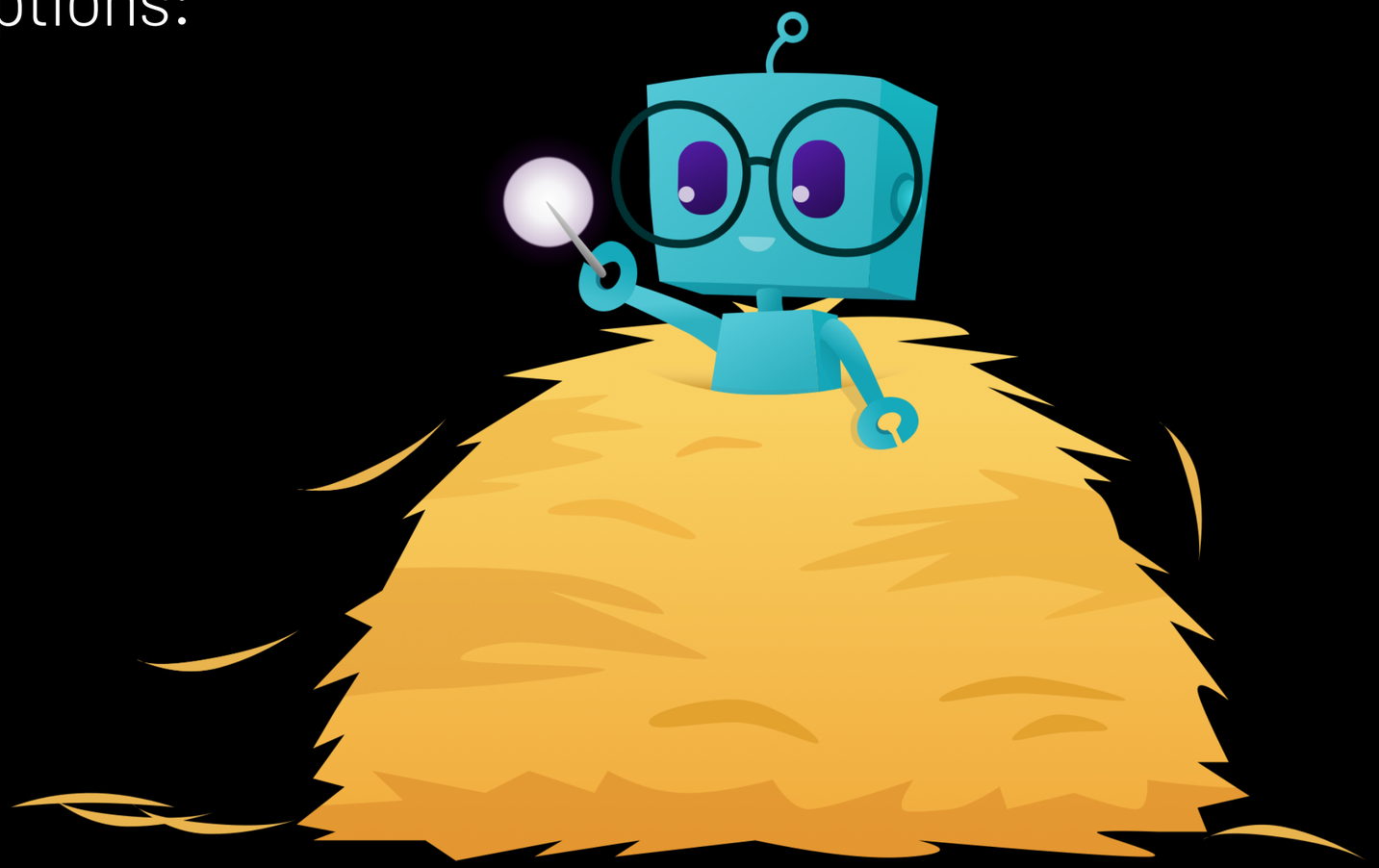
```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



Machine Learning Model Preparation

in our journy to find the best model that will give us the accurate answer for our question, we have few options:

1. Logistic Regression
2. Random forest
3. KNN
4. Desicion tree



Machine Learning Models

```
score = round(lr.score(x_test, y_test)*100, 2)
model.append("Logistic Regression")
acc.append(score)
print("Logistic Regression model accuracy is: ",
```

```
Logistic Regression model accuracy is:  65.71 %
```

```
score = round(rf.score(x_test, y_test)*100, 2)
model.append("Random Forest")
acc.append(score)
print("Random Forest model accuracy is: ", scor
```

```
Random Forest model accuracy is:  89.01 %
```

here all our 4 models and their accurate levels. now we have to compare all of them to see which one is the best for us....

```
score = round(knn.score(x_test, y_test)*100, 2)
model.append("KNeighbors Classifier")
acc.append(score)
print("KNeighbors Classifier model accuracy is: ",
```

```
KNeighbors Classifier model accuracy is:  78.35 %
```

```
score = round(dtc.score(x_test, y_test)*100, 2)
model.append("Decision Tree Classifier")
acc.append(score)
print("Decision Tree classifier model accuracy is: "
```

```
Decision Tree classifier model accuracy is:  86.37 %
```

Models Comparison

Models Comparison

```
comp = pd.DataFrame({"Model": model, "Accuracy[%]": acc})
comp
```

4]:

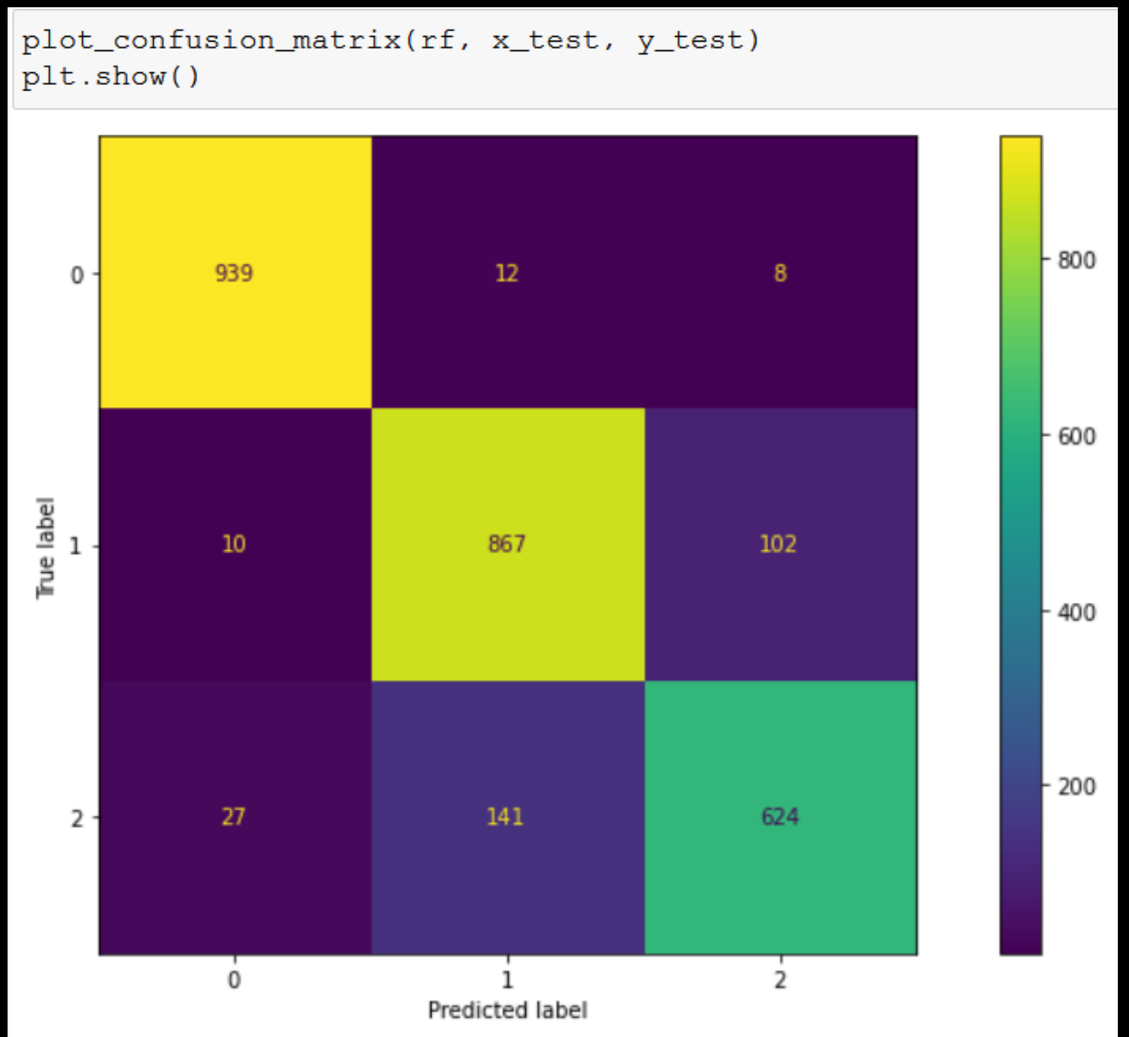
	Model	Accuracy[%]
0	Logistic Regression	65.71
1	Random Forest	89.01
2	KNeighbors Classifier	78.35
3	Decision Tree Classifier	86.37

As we can see, the Randon Forest model has the highest accurate percent, hence we will choose him to be our prediction model.

Finalized Model

So now when we have our chosen model,we will use confusion matrix to ensure that our predictions are true

```
plot_confusion_matrix(rf, x_test, y_test)
plt.show()
```



Prediction

Now when we have the model and its conclusions we can predict what will be status of the train. here's an example:

```
# Function to make prediction
def predict(data):
    data = np.reshape(data, (1, -1))
    res = rf.predict(data)

    if res[0]==0:
        return "In_Time"
    elif res[0]==1:
        return "Late"
    else:
        return "Precede"
```

```
# Making prediction for new data
data = x_test.iloc[0].values
res = predict(data)
print(f"Train that having data {data} will be {res}")
```

```
Train that having data [2016.    5.   29.   63.] will be Late
```

Conclusion

we found our CSV(was too hard to get), cleaned the broken and missing data from the data frame we created, started with EDA process, used few machine learning methods, according to the stages we have done - Random Forest gave us the best accuracy that is almost 89%. After we predicted that a specific train will leave the station "Late, In-time, Precede" and the model predicted if the train will leave late. So depending on this prediction we can answer the question will the train i'm heading for, will leave the train station at time?

Thank you!