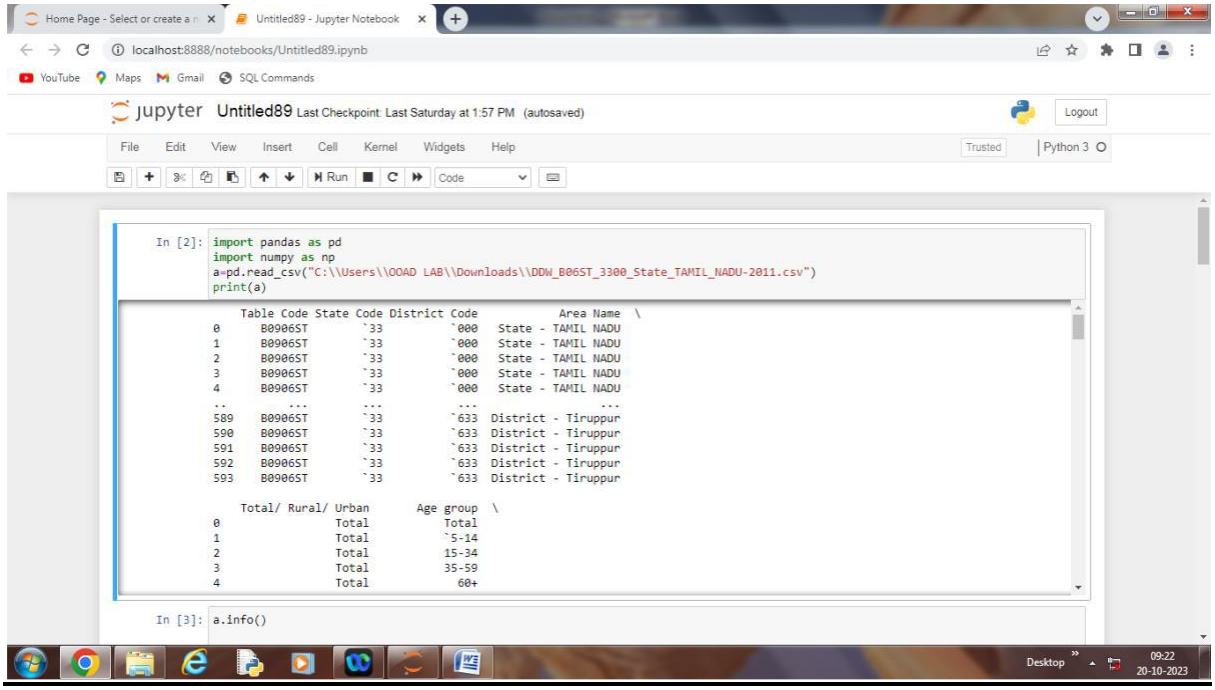
PHASE- IV

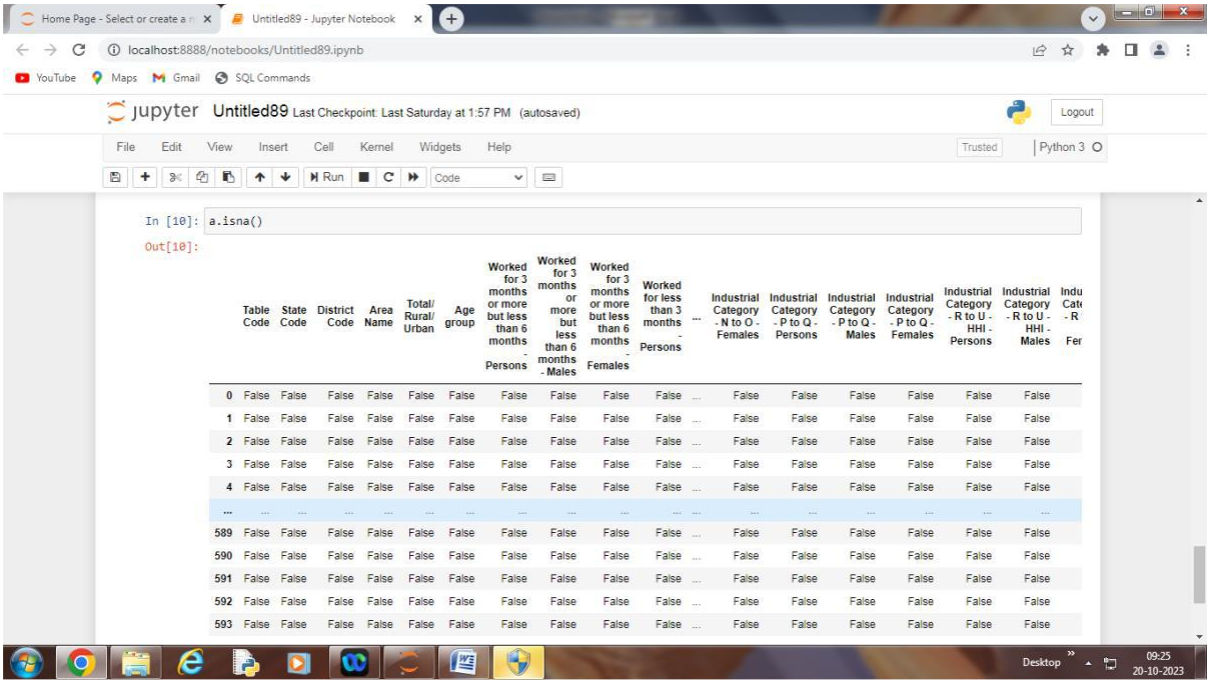
STOCK PRICE PREDICTION

DATA PREPROCESSING

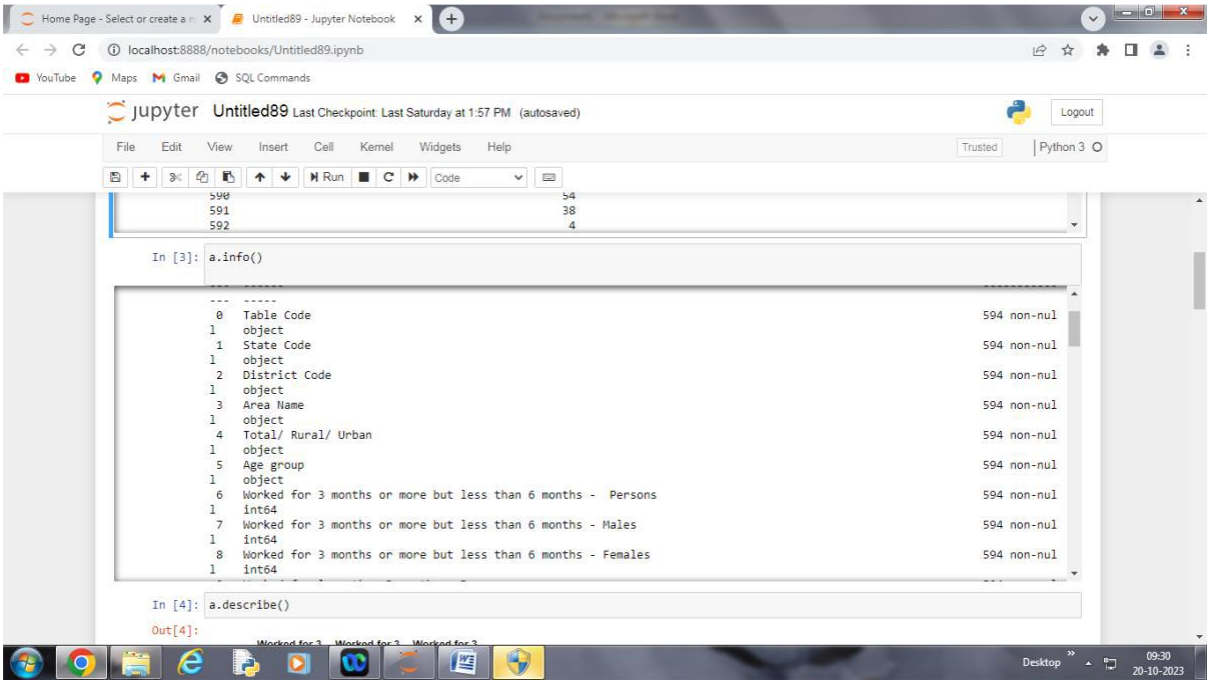
- **LOADING A DATASETS**

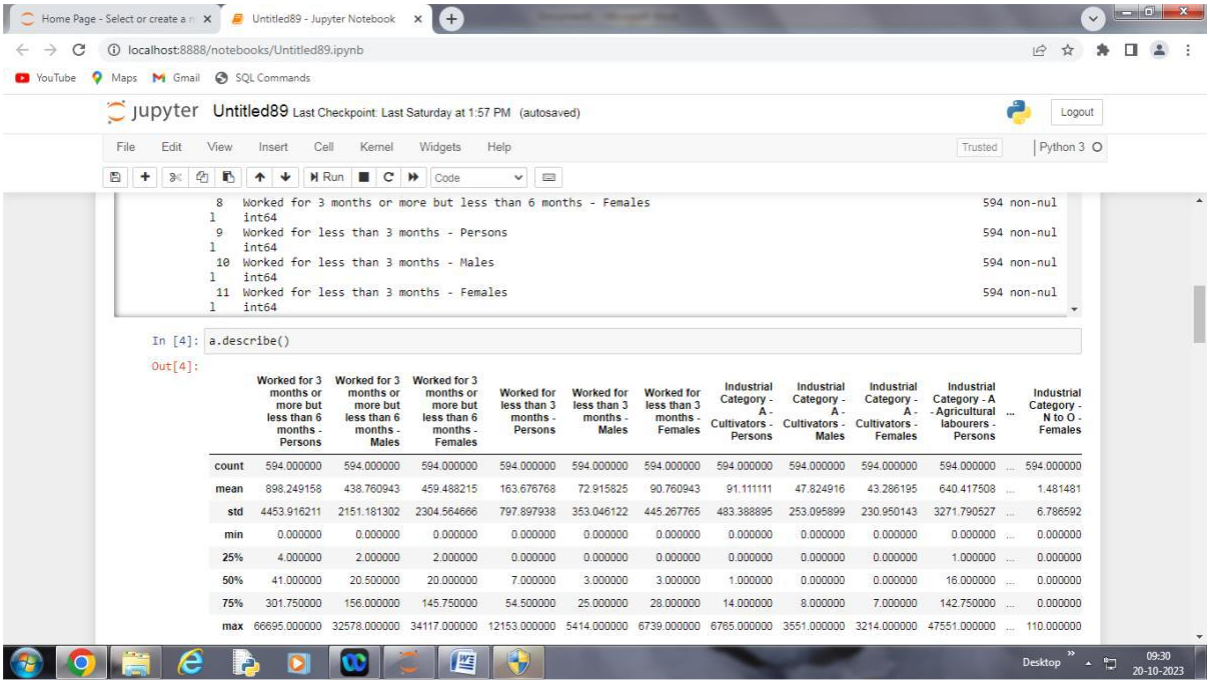


- ***PREPROCESSING THE DATASETS***



➢ *PREFROMING THE DIFFERENT ANALYSIS*





```
 8   Worked for 3 months or more but less than 6 months - Females   594 non-nul
l    int64
 9   Worked for less than 3 months - Persons                        594 non-nul
l    int64
 10  Worked for less than 3 months - Males                          594 non-nul
l    int64
 11  Worked for less than 3 months - Females                        594 non-nul
l    int64
```

In [4]:
```
a.describe()
```

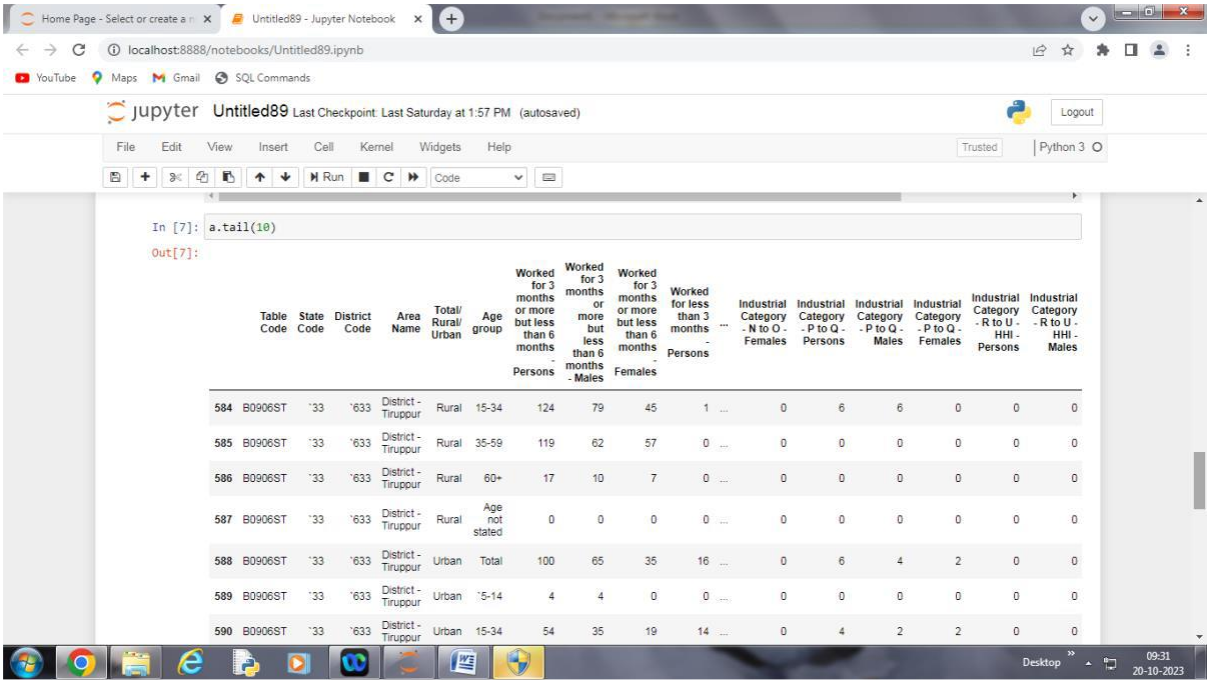
Out[4]:

	Worked for 3 months or more but less than 6 months - Persons	Worked for 3 months or more but less than 6 months - Males	Worked for 3 months or more but less than 6 months - Females	Worked for less than 3 months - Persons	Worked for less than 3 months - Males	Worked for less than 3 months - Females	Industrial Category - A - Cultivators - Persons	Industrial Category - A - Cultivators - Males	Industrial Category - A - Cultivators - Females	Industrial Category - A - Agricultural labourers - Persons	...	Industrial Category - N to O - Females
count	594.000000	594.000000	594.000000	594.000000	594.000000	594.000000	594.000000	594.000000	594.000000	594.000000	...	594.000000
mean	898.249158	438.760943	459.488215	163.676768	72.915825	90.760943	91.111111	47.824916	43.286195	640.417508	...	1.481481
std	4453.916211	2151.181302	2304.564666	797.897938	353.046122	445.267765	483.388895	253.095899	230.950143	3271.790527	...	6.786592
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
25%	4.000000	2.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	...	0.000000
50%	41.000000	20.500000	20.000000	7.000000	3.000000	3.000000	1.000000	0.000000	0.000000	16.000000	...	0.000000
75%	301.750000	156.000000	145.750000	54.500000	25.000000	28.000000	14.000000	8.000000	7.000000	142.750000	...	0.000000
max	66695.000000	32578.000000	34117.000000	12153.000000	5414.000000	6739.000000	6765.000000	3551.000000	3214.000000	47551.000000	...	110.000000

In [6]:
```
a.head(10)
```

Out[6]:

	Table Code	State Code	District Code	Area Name	Total/ Rural/ Urban	Age group	Worked for 3 months or more but less than 6 months - Persons	Worked for 3 months or more but less than 6 months - Males	Worked for 3 months or more but less than 6 months - Females	Worked for less than 3 months - Persons	...	Industrial Category - N to O - Females	Industrial Category - P to Q - Persons	Industrial Category - P to Q - Males	Industrial Category - P to Q - Females	Industrial Category - R to U - HHI - Persons	Industrial Category - R to U - HHI - Males	Ind Ca - F Fe
0	B0906ST	'33	'000	State - TAMIL NADU	Total	Total	66695	32578	34117	12153	...	110	278	128	150	978	226	
1	B0906ST	'33	'000	State - TAMIL NADU	Total	'5-14	2637	1345	1292	356	...	0	14	6	8	36	16	
2	B0906ST	'33	'000	State - TAMIL NADU	Total	15-34	31370	15374	15996	5714	...	46	198	94	104	508	114	
3	B0906ST	'33	'000	State - TAMIL NADU	Total	35-59	27418	12976	14442	4757	...	52	60	24	36	356	68	
				State														



Home Page - Select or create a n | Untitled89 - Jupyter Notebook

localhost:8888/notebooks/Untitled89.ipynb

YouTube Maps Gmail SQL Commands

jupyter Untitled89 Last Checkpoint: Last Saturday at 1:57 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

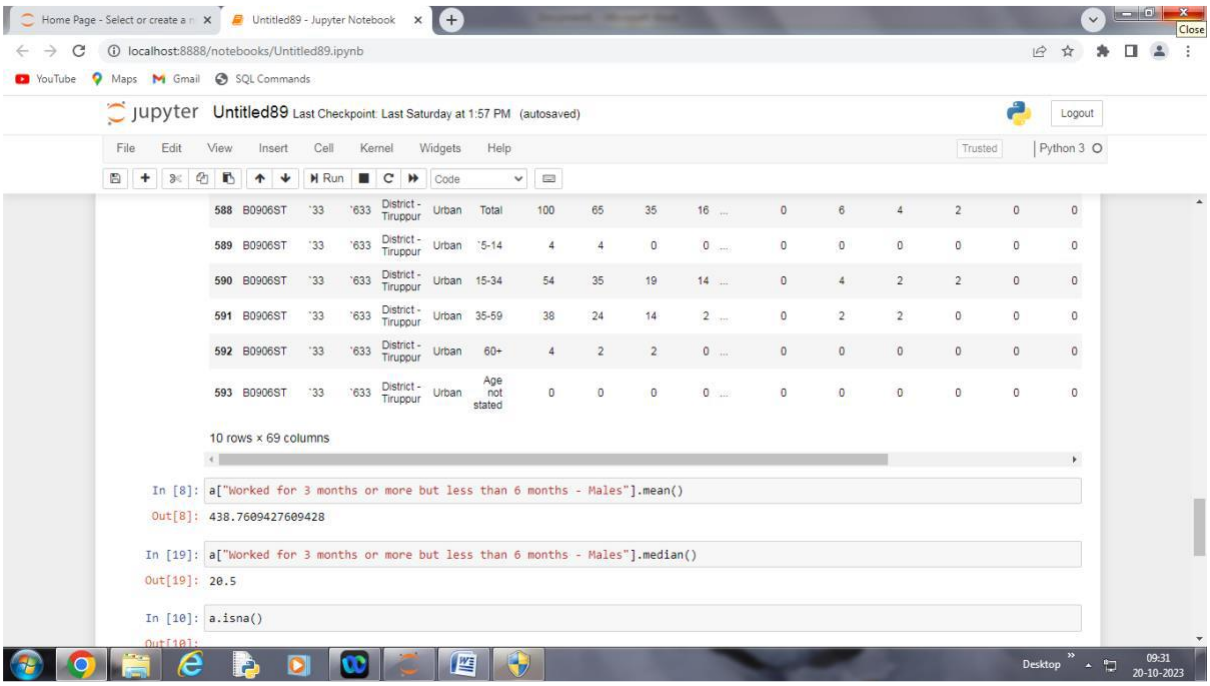
Run Code

In [7]: a.tail(10)

Out[7]:

	Table Code	State Code	District Code	Area Name	Total/ Rural/ Urban	Age group	Worked for 3 months or more but less than 6 months - Persons	Worked for 3 months or more but less than 6 months - Males	Worked for 3 months or more but less than 6 months - Females	Worked for less than 3 months - Persons	...	Industrial Category - N to O - Females	Industrial Category - P to Q - Persons	Industrial Category - P to Q - Males	Industrial Category - P to Q - Females	Industrial Category - R to U - HHI - Persons	Industrial Category - R to U - HHI - Males
584	B0906ST	'33	'633	District - Tiruppur	Rural	15-34	124	79	45	1	...	0	6	6	0	0	0
585	B0906ST	'33	'633	District - Tiruppur	Rural	35-59	119	62	57	0	...	0	0	0	0	0	0
586	B0906ST	'33	'633	District - Tiruppur	Rural	60+	17	10	7	0	...	0	0	0	0	0	0
587	B0906ST	'33	'633	District - Tiruppur	Rural	Age not stated	0	0	0	0	...	0	0	0	0	0	0
588	B0906ST	'33	'633	District - Tiruppur	Urban	Total	100	65	35	16	...	0	6	4	2	0	0
589	B0906ST	'33	'633	District - Tiruppur	Urban	'5-14	4	4	0	0	...	0	0	0	0	0	0
590	B0906ST	'33	'633	District - Tiruppur	Urban	15-34	54	35	19	14	...	0	4	2	2	0	0

Desktop 09:31 20-10-2023



Home Page - Select or create a n | Untitled89 - Jupyter Notebook

localhost:8888/notebooks/Untitled89.ipynb

YouTube Maps Gmail SQL Commands

jupyter Untitled89 Last Checkpoint: Last Saturday at 1:57 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

588	B0906ST	'33	'633	District - Tiruppur	Urban	Total	100	65	35	16	...	0	6	4	2	0	0
589	B0906ST	'33	'633	District - Tiruppur	Urban	'5-14	4	4	0	0	...	0	0	0	0	0	0
590	B0906ST	'33	'633	District - Tiruppur	Urban	15-34	54	35	19	14	...	0	4	2	2	0	0
591	B0906ST	'33	'633	District - Tiruppur	Urban	35-59	38	24	14	2	...	0	2	2	0	0	0
592	B0906ST	'33	'633	District - Tiruppur	Urban	60+	4	2	2	0	...	0	0	0	0	0	0
593	B0906ST	'33	'633	District - Tiruppur	Urban	Age not stated	0	0	0	0	...	0	0	0	0	0	0

10 rows × 69 columns

In [8]: a["Worked for 3 months or more but less than 6 months - Males"].mean()

Out[8]: 438.7609427609428

In [19]: a["Worked for 3 months or more but less than 6 months - Males"].median()

Out[19]: 20.5

In [10]: a.isna()

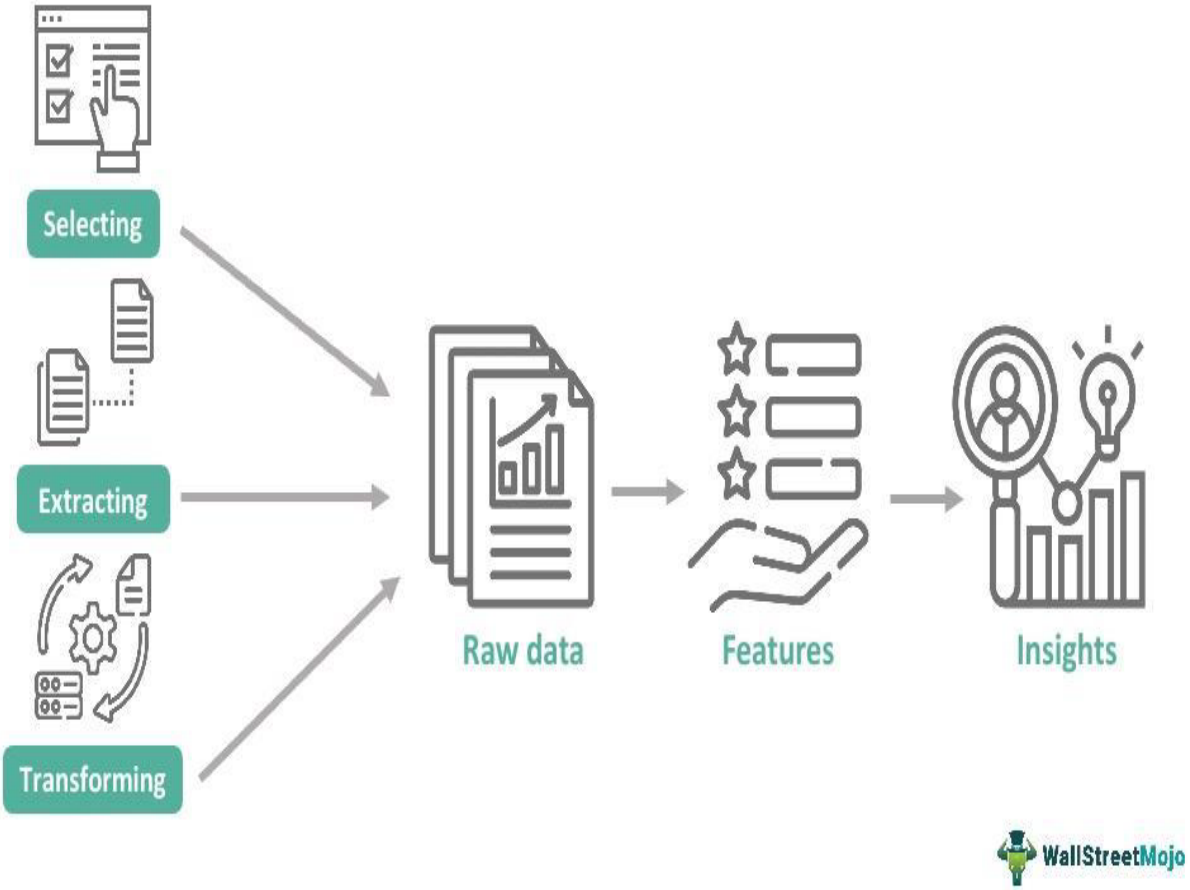
Out[10]:

Desktop 09:31 20-10-2023

FEATURE ENGINEERING

Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling.

What is Feature Engineering?



```
# Number of contractions (can't, won't, don't, haven't, etc.) in text
import re

def contraction_count(sent):
    count = 0
    count += re.subn(r"won\'t", '', sent)[1]
    count += re.subn(r"can\'t", '', sent)[1]
    count += re.subn(r"n\'t", '', sent)[1]
    count += re.subn(r"\'re", '', sent)[1]
    count += re.subn(r"\'s", '', sent)[1]
    count += re.subn(r"\'d", '', sent)[1]
    count += re.subn(r"\'ll", '', sent)[1]
    count += re.subn(r"\'t", '', sent)[1]
    count += re.subn(r"\'ve", '', sent)[1]
    count += re.subn(r"\'m", '', sent)[1]
    return count


df["excerpt_num_contractions"] = df["excerpt"].apply(contraction_count)
df[["excerpt","excerpt_num_contractions"]].head()
```

	excerpt	excerpt_num_contractions
2089	Alice looked at the jury-box, and saw that, in...	0
2806	Artificial intelligence (AI) is intelligence e...	0
1146	A gruff squire on horseback with shiny top boo...	0
1110	But that hadn't helped Washington.\nThe Americ...	2
196	The principal business of the people of this c...	0

WHAT IS MODEL TRAINING?

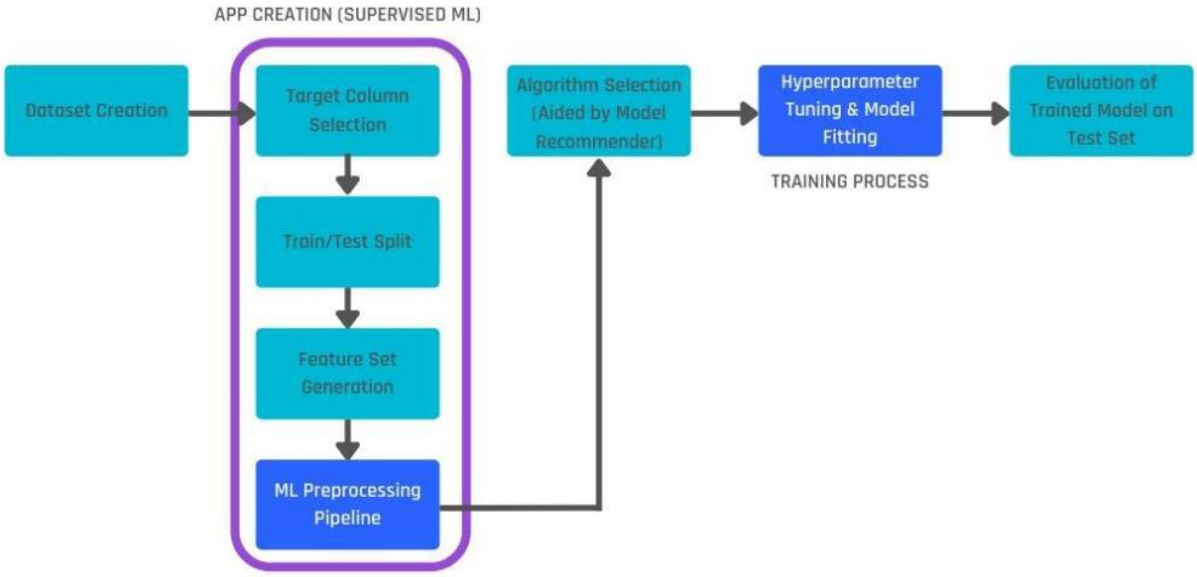
Model training is the phase in the data science development lifecycle where practitioners try to fit the best combination of weights and bias to a machine learning algorithm to minimize a loss function over the prediction range.

STEPS TO TRAINING MACHINE LEARNING MODEL

•Step 1: Begin with existing data. Machine learning requires us to have existing data—not the data our application will use when we run it, but data to learn from. ...

•Step 2: Analyze data to identify patterns. ...

•Step 3: Make predictions




```
In [10]: # prepare data frame for splitting data into train and test datasets

features = []
features = df_churn_pd.drop(['CHURNRISK'], axis=1)

label_churn = pd.DataFrame(df_churn_pd, columns = ['CHURNRISK'])
label_encoder = LabelEncoder()
label = df_churn_pd['CHURNRISK']

label = label_encoder.fit_transform(label)
print("Encoded value of Churnrisk after applying label encoder : " + str(label))
```

```
Encoded value of Churnrisk after applying label encoder : [2 1 1 ... 2 1 1]
```

MODEL EVALUATION

Model evaluation is a crucial aspect of machine learning, allowing us to assess how well our models perform on unseen data. In this step-by-step

guide, we will explore the process of model evaluation using Python. By following these steps and leveraging Python's powerful libraries, you'll gain valuable insights into your model's performance and be able to make informed decisions. Let's dive in and evaluate our machine learning models!

Step 1: Prepare the Data

The first step in model evaluation is to prepare your data. Split your dataset into training and test sets using the train_test_split function from the scikit-learn library. This ensures that we have separate data for training and evaluating our model.

```
from sklearn.model_selection import train_test_split
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Step 2: Train the Model
Next, select an appropriate model for your task and train it using the training set. For example, let's train a logistic regression model using scikit-learn:

```
from sklearn.linear_model import LogisticRegression
# Create an instance of the model
model = LogisticRegression()
# Train the model
model.fit(X_train, y_train)
```

Step 3: Evaluate on the Test Set
Now, it's time to evaluate our model on the test set. Use the trained model to make predictions on the test data and compare them to the actual labels. Calculate evaluation metrics such as `accuracy_score` to measure the model's performance.

```
from sklearn.metrics import accuracy_score
# Make predictions on the test set
y_pred = model.predict(X_test)
# Calculate accuracy
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Step 4: Perform Cross-Validation (Optional)
To obtain a more robust evaluation, you can perform cross-validation. This technique involves splitting the data into multiple folds and training/evaluating the model on different combinations. Here's an example using `cross_val_score` from scikit-learn:

```
from sklearn.model_selection import cross_val_score
# Perform cross-validation
scores = cross_val_score(model, X, y, cv=5)
# Calculate the average performance across all folds
mean_accuracy = scores.mean()
print("Mean Accuracy:", mean_accuracy)
```

Step 5: Assess Model's Performance

Analyze the evaluation metrics obtained from the previous steps to assess the model's performance. Consider the context of your problem and compare the results against your desired performance level or any baseline models. This analysis will provide insights into the strengths and weaknesses of your model.

Step 6: Iterate and Improve (if needed)

Based on the assessment, you may need to iterate and improve your model. Consider collecting more data, refining features, trying different algorithms, or tuning hyperparameters. Repeat the evaluation process until you achieve the desired performance.