```
import numpy as np
import pandas as pd
```

```
#For uplading files to google colaboratory
from google.colab import files
uploaded =files.upload()
# Remove the 'encoding' argument
rawdata = pd.read_excel('organic_keywords_dataset.xlsx')
# pandas will automatically detect the encoding in most cases.

#If you encounter encoding issues, try using the engine parameter for excel reader engine
# rawdata = pd.read_excel('organic_keywords_dataset.xlsx', engine='openpyxl')
```

Choose Files organic_ke…dataset.xlsx

- **organic_keywords_dataset.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 779930 bytes, last modified: 1/10/2025 - 100% done

```
Saving organic_keywords_dataset.xlsx to organic_keywords_dataset (3).xlsx
```

```
rawdata[:7]
```

	Keyword	Position	Previous position	Search Volume	Keyword Difficulty	CPC	URL	Traffic	Traffic (%)	Traffic Cost	Competiti
0	shower pan	3	3	33100	77.34	1.00	https://bestbath.com/products/showers/pans/	2979	14.03	2979	1
1	best bath	1	1	3600	78.95	1.85	https://bestbath.com/	2880	13.56	5328	1
2	shower base	5	5	22200	76.92	0.93	https://bestbath.com/products/showers/pans/	1110	5.22	1032	1
3	bathtub surround kits	1	1	1600	81.20	0.98	https://bestbath.com/products/tub-surrounds/	752	3.54	736	1
4	tub shower combo	5	5	14800	84.20	0.89	https://bestbath.com/products/tubs/shower-tub-...	740	3.48	658	1
5	handicap showers	3	3	5400	63.13	2.00	https://bestbath.com/products/showers/	486	2.28	972	1
6	tub surround	7	7	12100	83.42	0.99	https://bestbath.com/products/tub-surrounds/	484	2.28	479	1

```
# Define Features for training and testing of Ml models.
features = ["Traffic",
            "CPC",
            "Number of Results",
            "Search Volume"]
# Define Target (What should be predicted).
target = "Position"
#we can change both the features and target according to requirement.
```

```
# Split the datset in ratio 0f 4:1(80%:20%) for train and test data
train = rawdata.sample(frac=0.8)
test = rawdata.loc[~rawdata.index.isin(train.index)]
print ("Train rows: {}".format(len(train.index)))
print ("Test rows: {}".format(len(test.index)))
```

```
Train rows: 6067
Test rows: 1517
```

```
# Import different Ml Alorithms to see differences between their predictions
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
```

```
# The Function to print readable scores of the models
def print_scores(scores):
    r = 1
    for score in scores:
        print("Run: {} - Score: {}".format(r, score))
        r += 1
```

```
LinearRegressionModel =LinearRegression()
```

```
LinearRegressionModel.fit(train[features], train[target])
```

```
▾ LinearRegression
LinearRegression()
```

```
# Test how the model performes against the Training data we split above...
prediction_score = LinearRegressionModel.score(test[features], test[target])
print("The score of prediction for LinearRegressionModel is: {}".format(prediction_score))
```

```
The score of prediction for LinearRegressionModel is: 0.007383242533890666
```

```
DecisionTreeClassifierModel = DecisionTreeClassifier()
```

```
# Define RandaomForest Regressor model

pipeline = make_pipeline(preprocessing.StandardScaler(),
                         RandomForestRegressor(n_estimators=200))

# Declare hyperparameters to tune
hyperparameters = { 'randomforestregressor__max_features' : ['auto', 'sqrt', 'log2'],
                  'randomforestregressor__max_depth': [5, 3]}

# Tune model using cross-validation pipeline
RandomForestRegressorModel = GridSearchCV(pipeline, hyperparameters, cv=5)

RandomForestRegressorModel.fit(train[features], train[target])
prediction_score = RandomForestRegressorModel.score(test[features], test[target])
print("The score of prediction for RandomForestRegressorModel is: {}".format(prediction_score))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:528: FitFailedWarning:
10 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 1389, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/pipeline.py", line 660, in fit
    self._final_estimator.fit(Xt, y, **last_step_params["fit"])
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 1382, in wrapper
    estimator._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 436, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 98, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestRegressor must be an int in the r

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:1107: UserWarning: One or more of the test scores are non
  warnings.warn(
The score of prediction for RandomForestRegressorModel is: 0.299176265546371
```

```
# Import different Ml Alorithms to see differences between their predictions
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
# %%
# The Function to print readable scores of the models
```

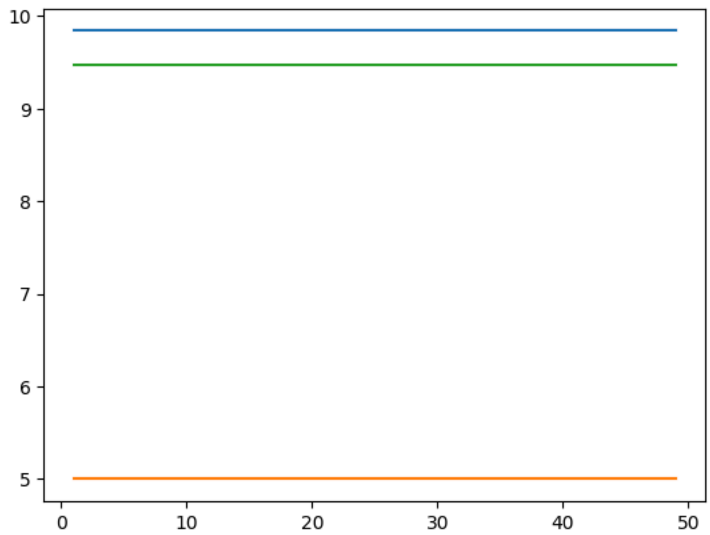
```
def print_scores(scores):
    r = 1
    for score in scores:
        print("Run: {} - Score: {}".format(r, score))
        r += 1
# %%
LinearRegressionModel =LinearRegression()
LinearRegressionModel.fit(train[features], train[target])
# %%
# Test how the model performes against the Training data we split above...
prediction_score = LinearRegressionModel.score(test[features], test[target])
print("The score of prediction for LinearRegressionModel is: {}".format(prediction_score))

# %%
DecisionTreeClassifierModel = DecisionTreeClassifier()

#Fit the DecisionTreeClassifierModel to the training data
DecisionTreeClassifierModel.fit(train[features], train[target]) # This line was missing

# %%
# Define RandaomForest Regressor model

pipeline = make_pipeline(preprocessing.StandardScaler(),
                         RandomForestRegressor(n_estimators=200))

# Declare hyperparameters to tune
hyperparameters = { 'randomforestregressor__max_features' : ['auto', 'sqrt', 'log2'],
                  'randomforestregressor__max_depth': [5, 3]}

# Tune model using cross-validation pipeline
RandomForestRegressorModel = GridSearchCV(pipeline, hyperparameters, cv=5)

RandomForestRegressorModel.fit(train[features], train[target])
prediction_score = RandomForestRegressorModel.score(test[features], test[target])
print("The score of prediction for RandomForestRegressorModel is: {}".format(prediction_score))
# %%
# Print Predictions for all created Models
# Give Sample values to parameters for predictions
sample = [[1032,0.93,469000000,22200]]  # needs to be same count as features

rawdata_to_predict = pd.DataFrame(data = sample, index=[0], columns=features)
result = LinearRegressionModel.predict(rawdata_to_predict)
print("LinearRegressionModel predicted:       {}".format(int(result[0])))
result = DecisionTreeClassifierModel.predict(rawdata_to_predict)
print("DecisionTreeClassifierModel predicted: {}".format(int(result[0])))
result = RandomForestRegressorModel.predict(rawdata_to_predict)
print("RandomForestRegressorModel predicted:  {}".format(int(result[0])))
```

```
The score of prediction for LinearRegressionModel is: 0.007383242533890666
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:528: FitFailedWarning:
10 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 1389, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/pipeline.py", line 660, in fit
    self._final_estimator.fit(Xt, y, **last_step_params["fit"])
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 1382, in wrapper
    estimator._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 436, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 98, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestRegressor must be an int in the r

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:1107: UserWarning: One or more of the test scores are nor
  warnings.warn(
The score of prediction for RandomForestRegressorModel is: 0.29812766394217016
LinearRegressionModel predicted:       9
DecisionTreeClassifierModel predicted: 5
RandomForestRegressorModel predicted:  9
```

```
#  Define the function to plot Models
import matplotlib.pyplot as plt
```

```
def plt_ctr_from_to_position(models, features, from_pos, to_pos, data):
    for model in models:
        predictions_x = []
        predictions_y = []
        positions = range(from_pos, to_pos)
        for pos in positions:
            df_to_predict = pd.DataFrame(data = sample, index=[0], columns=features)
            predictions_x.append(pos)
            predictions_y.append(model.predict(df_to_predict)[0])
        predictions_x, predictions_y
        plt.plot(predictions_x, predictions_y)
```

```
plt_ctr_from_to_position([LinearRegressionModel, DecisionTreeClassifierModel, RandomForestRegressorModel], features, 1, 50, rawdata_to_pr
# Changed 'data' to 'rawdata_to_predict' as it's the DataFrame used for predictions in the previous cell.
# You may want to use 'rawdata' instead if you want to visualize the whole dataset.
```



```
#Define the function such that you like to filter the data.
def analyzePositionSpecs(min_p,max_p):
    jk = rawdata.loc[(rawdata['Position'] >= min_p) & (rawdata['Position'] <= max_p)]
    return jk
```

```
#Give desired filter positions to function.
jk = analyzePositionSpecs(5,15)

# Creates the total volume for 'Search Volume' column
def Volume(x):
    total = x['Search Volume'].sum()
    return total
netvolume = Volume(jk)

# Converting search volume into a percentage
svp = (jk['Search Volume']/netvolume)*100
```

```
# Make booleans for filter parameters, change if statement for choice parameters
newData = []
for value in jk['Search Volume']:
    if (value/netvolume)*100 >= 1:
        newData.append(True)
    else:
        newData.append(False)
```

```
# add new columns variables to data frame
jk.loc[:,'Volume Ratio'] = svp
jk.loc[:,'Result']= newData
```

```
# create final dataframe
finalresult = jk.loc[jk['Result'] == True]
finalresult
```

```
<ipython-input-47-dd39ec6ec251>:27: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise in a future er
  jk.loc[:,'Result']= newData
```

	Keyword	Position	Previous position	Search Volume	Keyword Difficulty	CPC	URL	Traffic	Traffic (%)	Traffic Cost	Competiti
2	shower base	5	5	22200	76.92	0.93	https://bestbath.com/products/showers/pans/	1110	5.22	1032	
4	tub shower combo	5	5	14800	84.20	0.89	https://bestbath.com/products/tubs/shower-tub-...	740	3.48	658	
6	tub surround	7	7	12100	83.42	0.99	https://bestbath.com/products/tub-surrounds/	484	2.28	479	
8	shower tub	6	6	8100	77.51	1.04	https://bestbath.com/products/tubs/shower-tub-...	405	1.90	421	
9	bathtub shower combo	5	5	8100	85.40	0.81	https://bestbath.com/products/tubs/shower-tub-...	405	1.90	328	
11	bathtub surround	7	7	6600	84.19	1.19	https://bestbath.com/products/tub-surrounds/	264	1.24	314	
21	bathtub shower	8	8	4400	84.60	1.10	https://bestbath.com/products/tubs/shower-tub-...	132	0.62	145	
22	shower base sizes	5	5	2400	76.43	0.92	https://bestbath.com/products/showers/pans/	120	0.56	110	
23	tub and shower combo	7	7	2900	86.03	0.72	https://bestbath.com/products/tubs/shower-tub-...	116	0.54	83	
25	garden tub	14	14	14800	83.70	0.69	https://bestbath.com/products/tubs/garden-tubs/	103	0.48	71	
40	tub and shower	8	8	2400	80.86	0.87	https://bestbath.com/products/tubs/shower-tub-...	72	0.33	62	
139	custom shower pan	15	15	3600	59.63	1.17	https://bestbath.com/types-shower-pans/	18	0.08	21	

Next steps: Generate code with `finalresult` View recommended plots New interactive sheet