Prediction by Name

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score
```

```
import chardet

# Detect the encoding of the file
with open('/content/drive/MyDrive/Indian-Name-12.csv', 'rb') as f:
    result = chardet.detect(f.read())

# Print the detected encoding
print(result['encoding'])

# Read the CSV file with the detected encoding
df = pd.read_csv('/content/drive/MyDrive/Indian-Name-12.csv', encoding=result['encoding'])
```

```
Windows-1252
```

```
# lowercase name column
df["Name"] = df["Name"].str.lower()

# --- create new features ---

# add name_length column
df["name_length"] = df["Name"].str.len()

# add starts_with_{alphabet}, ends_with_{alphabet}, and count_{alphabet} columns
alphabet = "abcdefghijklmnopqrstuvwxyz"
for letter in alphabet:
    df[f"begins_with_{letter}"] = df["Name"].apply(lambda n: 1 if n[0] == letter else 0)
    df[f"ends_with_{letter}"] = df["Name"].apply(lambda n: 1 if n[-1] == letter else 0)
    df[f"count_{letter}"] = df["Name"].apply(lambda n: n.count(letter))
```

```
df.head()
```

	Name	Target	name_length	begins_with_a	ends_with_
0	yash	1	4	0	
1	prit	1	4	0	
2	meet	1	4	0	
3	drashti	0	7	0	
4	saloni	0	6	0	

5 rows × 81 columns

```
X = df.drop(["Name", "Target"], axis=1).to_numpy()
y = df["Target"].to_numpy()

n = X.shape[1]
```

```
print(f"X: {X.shape}")
print(f"y: {y.shape}")
```

```
X: (1299, 79)
y: (1299,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

```
print(f"Training data:  X_train: {X_train.shape}, y_train: {y_train.shape}")
print(f"Test data:      X_test:  {X_test.shape}, y_test:  {y_test.shape}")
```

```
Training data:  X_train: (974, 79), y_train: (974,)
Test data:      X_test:  (325, 79), y_test:  (325,)
```

```
X_train[:5, :5]
```

```
array([[5, 1, 1, 2, 0],
       [6, 0, 0, 0, 0],
       [6, 0, 1, 1, 0],
       [5, 1, 0, 2, 0],
       [6, 0, 0, 1, 0]])
```

```
scaler = StandardScaler().fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
X_train[:5, :5]
```

```
array([[-0.60963755,  2.9563491 ,  1.5395591 ,  1.07308958, -0.18431027],
       [ 0.11010424, -0.33825505, -0.64953661, -1.40954523, -0.18431027],
       [ 0.11010424, -0.33825505,  1.5395591 , -0.16822782, -0.18431027],
       [-0.60963755,  2.9563491 , -0.64953661,  1.07308958, -0.18431027],
       [ 0.11010424, -0.33825505, -0.64953661, -0.16822782, -0.18431027]])
```

1. Logistic Regression

```
clf = LogisticRegression().fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)

acc = (np.sum(y_pred == y_test) / y_test.size) * 100
print(f"The accuracy of the model on the test set is {round(acc, 2)}%.")
```

```
The accuracy of the model on the test set is 84.62%.
```

```
def predict_gender(name):

    name = name.lower()

    # store features
    x = []



    # add name_length
    x.append(len(name))


    alphabet = "abcdefghijklmnopqrstuvwxyz"
    for letter in alphabet:
        x.append(1 if name[0] == letter else 0)
        x.append(1 if name[-1] == letter else 0)
        x.append(name.count(letter))

    # convert feature vector to a numpy array
    x = np.asarray(x).reshape(1, -1)

    # scale feature vector
    x = scaler.transform(x)

    # make prediction
    pred = clf.predict(x)[0]
    pred = "Male" if pred == 1 else "Female"

    return pred

name = "Nilesh"
pred = predict_gender(name)
print(f"Predicted gender 1: {pred}")
```

```
Predicted gender 1: Male
```

2. SVM

```
svm_clf = SVC().fit(X_train, y_train)
svm_pred = svm_clf.predict(X_test)
svm_acc = accuracy_score(y_test, svm_pred)
print(f"SVM Accuracy: {round(svm_acc * 100, 2)}%")
```

```
SVM Accuracy: 86.15%
```

```
def predict_gender(name):

    name = name.lower()

    # store features
    x = []



    # add name_length
    x.append(len(name))


    alphabet = "abcdefghijklmnopqrstuvwxyz"
    for letter in alphabet:
        x.append(1 if name[0] == letter else 0)
        x.append(1 if name[-1] == letter else 0)
        x.append(name.count(letter))

    # convert feature vector to a numpy array
    x = np.asarray(x).reshape(1, -1)

    # scale feature vector
    x = scaler.transform(x)

    # make prediction
    pred = svm_clf.predict(x)[0]
    pred = "Male" if pred == 1 else "Female"

    return pred

name = "Nilesh"
pred = predict_gender(name)
print(f"Predicted gender 2: {pred}")
```

```
Predicted gender 2: Male
```

3. KNN

```
knn_clf = KNeighborsClassifier().fit(X_train, y_train)
knn_pred = knn_clf.predict(X_test)
knn_acc = accuracy_score(y_test, knn_pred)
print(f"KNN Accuracy: {round(knn_acc * 100, 2)}%")
```

```
KNN Accuracy: 75.38%
```

```
def predict_gender(name):

    name = name.lower()

    # store features
    x = []



    # add name_length
    x.append(len(name))


    alphabet = "abcdefghijklmnopqrstuvwxyz"
    for letter in alphabet:
        x.append(1 if name[0] == letter else 0)
        x.append(1 if name[-1] == letter else 0)
        x.append(name.count(letter))

    # convert feature vector to a numpy array
    x = np.asarray(x).reshape(1, -1)

    # scale feature vector
    x = scaler.transform(x)

    # make prediction
    pred = knn_clf.predict(x)[0]
    pred = "Male" if pred == 1 else "Female"

    return pred

name = "Nilesh"
pred = predict_gender(name)
print(f"Predicted gender 3: {pred}")
```

```
Predicted gender 3: Male
```

4.Dissision Tree Classifier

```
dt_clf = DecisionTreeClassifier().fit(X_train, y_train)
dt_pred = dt_clf.predict(X_test)
dt_acc = accuracy_score(y_test, dt_pred)
print(f"Decision Tree Accuracy: {round(dt_acc * 100, 2)}%")
```

```
Decision Tree Accuracy: 83.38%
```

```
def predict_gender(name):

    name = name.lower()

    # store features
    x = []



    # add name_length
    x.append(len(name))


    alphabet = "abcdefghijklmnopqrstuvwxyz"
    for letter in alphabet:
        x.append(1 if name[0] == letter else 0)
        x.append(1 if name[-1] == letter else 0)
        x.append(name.count(letter))

    # convert feature vector to a numpy array
    x = np.asarray(x).reshape(1, -1)

    # scale feature vector
    x = scaler.transform(x)

    # make prediction
    pred = dt_clf.predict(x)[0]
    pred = "Male" if pred == 1 else "Female"

    return pred

name = "Nilesh"
pred = predict_gender(name)
print(f"Predicted gender 4 : {pred}")
```

```
Predicted gender 4 : Male
```

5. Neural Network

```
# Make sure to scale the input features for better performance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define a simple Neural Network
nn_model = MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000)
nn_model.fit(X_train_scaled, y_train)
nn_predictions = nn_model.predict(X_test_scaled)
nn_accuracy = accuracy_score(y_test, nn_predictions)
print(f"Neural Network Accuracy: {round(nn_accuracy* 100, 2)}%")
```

```
Neural Network Accuracy: 87.38%
```

```
def predict_gender(name):

    name = name.lower()
```

```
    # store features
    x = []

    # add name_length
    x.append(len(name))

    alphabet = "abcdefghijklmnopqrstuvwxyz"
    for letter in alphabet:
        x.append(1 if name[0] == letter else 0)
        x.append(1 if name[-1] == letter else 0)
        x.append(name.count(letter))

    # convert feature vector to a numpy array
    x = np.asarray(x).reshape(1, -1)

    # scale feature vector
    x = scaler.transform(x)

    # make prediction
    pred = nn_model.predict(x)[0]
    pred = "Male" if pred == 1 else "Female"

    return pred

name = "Nilesh"
pred = predict_gender(name)
print(f"Predicted gender 5 : {pred}")
```

```
Predicted gender 5 : Male
```