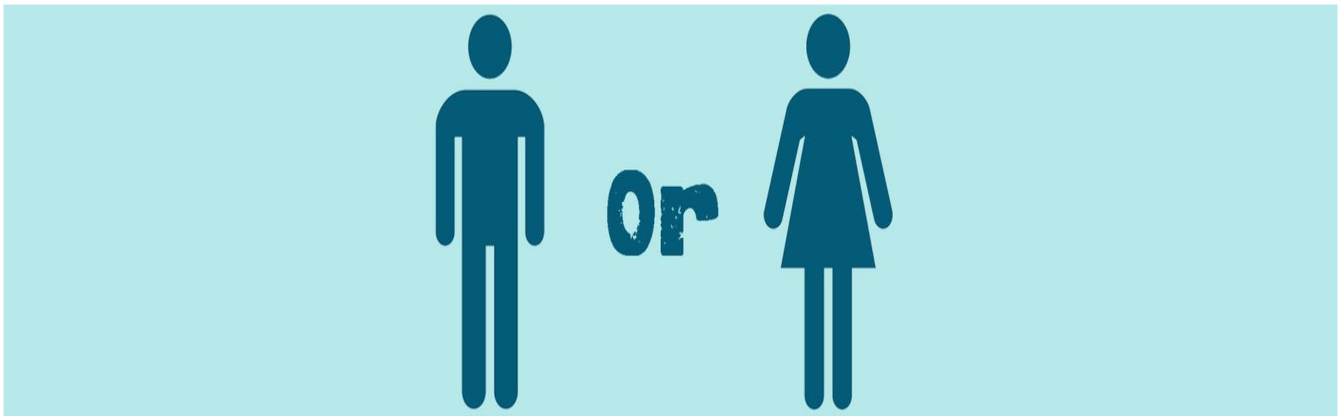
Title- Gender Identification.

Submitted By
KEDAR RAVINDRA KALOKHE (EBEON0622606756)



Objective:-

Identification of Gender whether it's Male or Female based on some parameters.

Study of Existing System:-

Reference has been taken from Kaggle Website – Name of the Dataset – Gender Classification Dataset By Solved By DANIEL FOURIE.

Gaps in Existing System:-

- He has performed a few Eda(Exploratory Data Analysis) visualization on Dataset.
- And he has just performed KNN(K-Nearest Algorithm) to Predict.

Proposed Solution:-

- Performed more Machine learning Algorithms to get more predictability.
- Performed more Eda(Exploratory Data Analysis) visualization on Dataset.

Features & Predictor:-

This dataset contains 7 features and a label column.

1. $\mathrm{long}hair$ - This column contains 0's and 1's where 1 is "long hair" and 0 is "not long hair".
2. $\mathrm{foreheadwidth}cm$ - This column is in Cm. This is the width of the forehead.
3. $\mathrm{foreheadheight}cm$ - This is the height of the forehead and it's in Cm.

4. nosewide - This column contains 0's and 1's where 1 is "wide nose" and 0 is "not wide nose".
5. nose*long* - This column contains 0's and 1's where 1 is "Long nose" and 0 is "not long nose".
6. lipsthin - This column contains 0's and 1's where 1 represents the "thin lips" while 0 is "Not thin lips".
7. distance*nose*to*lip*long - This column contains 0's and 1's where 1 represents the "long distance between nose and lips" while 0 is "short distance between nose and lips".

8. gender - This is either "Male" or "Female"

Tools/Technology used to implement Proposed Solution:-

- Python
- Pandas
- Numpy
- Matplotlib

- Seaborn
- Excel

In Machine learning Algorithms Following are Used:-

1. **Logistic Regression**:

Logistic regression is often used a lot of times in machine learning for predicting the likelihood of response attributes when a set of explanatory independent attributes are given. It is used when the target attribute is also known as a dependent variable with categorical values like yes/no, true/false, etc. It’s widely used for solving classification problems. It falls under the category of supervised machine learning. It efficiently solves linear and 12 binary classification problems. It is one of the most commonly used and easy-to-implement algorithms. It’s a statistical technique to predict binary classes. When the target variable has two possible classes, it predicts the likelihood of the event's occurrence. In our dataset, the target variable is categorical as it has only two classes-yes/no.

2. Decision Tree:

A decision tree is a non-parametric supervised learning algorithm utilized for classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes, and leaf nodes.

3. Random Forest :

Random Forest is the most famous and it is considered the best algorithm for machine learning. It is a supervised learning algorithm. To achieve more accurate and consistent prediction, a random forest creates several decision trees and combines them. The major benefit of using it is its ability to solve both regression and classification issues. When building each tree, it employs bagging and feature randomness to produce an uncorrelated tree forest whose collective forecast has much better accuracy than any individual tree’s prediction. Bagging enhances the accuracy of machine learning methods by grouping them. In this algorithm, during the splitting of nodes, it takes only a random subset of nodes into an account. When splitting a node, it looks for the best feature from a random group of features rather than the most significant feature. This results in getting better accuracy. It efficiently deals with huge datasets. It also solves the issue of overfitting in datasets. It works as follows: First, it'll select random samples from the provided dataset. Next, for every selected sample it'll create a decision tree and it'll receive a forecasted result from every created decision tree. Then for each result that was predicted, it'll perform voting and through voting, it will select the best-predicted result.

4.K Nearest Neighbor (KNN) :

KNN is a supervised machine learning algorithm. It assumes similar objects are nearer to one another. When the parameters are continuous in that case knn is preferred. This algorithm classifies objects by predicting their nearest neighbor. It's simple and easy to implement and also has high speed because of which it is preferred over the other algorithms when it comes to solving classification problems.

5.Naive Bayes :

It is a probabilistic machine learning algorithm that is mainly used in classification problems. 11 | Page It's based on the Bayes theorem. It is simple and easy to build. It deals with huge datasets efficiently. It can solve complicated classification problems. The existence of a specific feature in a class is assumed to be independent of the presence of any other feature according to naïve Bayes theorem. Its formula is as follows : P(S|T) = P(T|S) * P(S) / P(T) Here, T is the event to be predicted, and S is the class value for an event. This equation. will find out the class in which the expected feature is for classification.

Importing libraries

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Reading Data

```python
data=pd.read_csv(r"D:\kedar\csv files\gender_classification_v7.csv")
```

```python
data
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance
0	1	11.8	6.1	1	0	1	
1	0	14.0	5.4	0	0	1	
2	0	11.8	6.3	1	1	1	
3	0	14.4	6.1	0	1	1	
4	1	13.5	5.9	0	0	0	
...	...	...	...	...	...	...	
4996	1	13.6	5.1	0	0	0	
4997	1	11.9	5.4	0	0	0	
4998	1	12.9	5.7	0	0	0	

Column Names

```
]:  # name of columns present in datset
    data.columns
```

```
]: Index(['long_hair', 'forehead_width_cm', 'forehead_height_cm', 'nose_wide',
          'nose_long', 'lips_thin', 'distance_nose_to_lip_long', 'gender'],
         dtype='object')
```

Checking unique value

```
#Checking unique value in datset
data.nunique()
```

```
long_hair                     2
forehead_width_cm            42
forehead_height_cm           21
nose_wide                     2
nose_long                     2
lips_thin                     2
distance_nose_to_lip_long     2
gender                        2
dtype: int64
```

All Unique Values of all Columns

Top 5 Rows

```
:  data.head(5)
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_no
0	1	11.8	6.1	1	0	1	
1	0	14.0	5.4	0	0	1	
2	0	11.8	6.3	1	1	1	
3	0	14.4	6.1	0	1	1	
4	1	13.5	5.9	0	0	0	

Bottom 5 Rows

```
data.tail(5)
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance
4996	1	13.6	5.1	0	0	0	
4997	1	11.9	5.4	0	0	0	
4998	1	12.9	5.7	0	0	0	
4999	1	13.2	6.2	0	0	0	
5000	1	15.4	5.4	1	1	1	

Shape of Dataset

```
data.shape
```

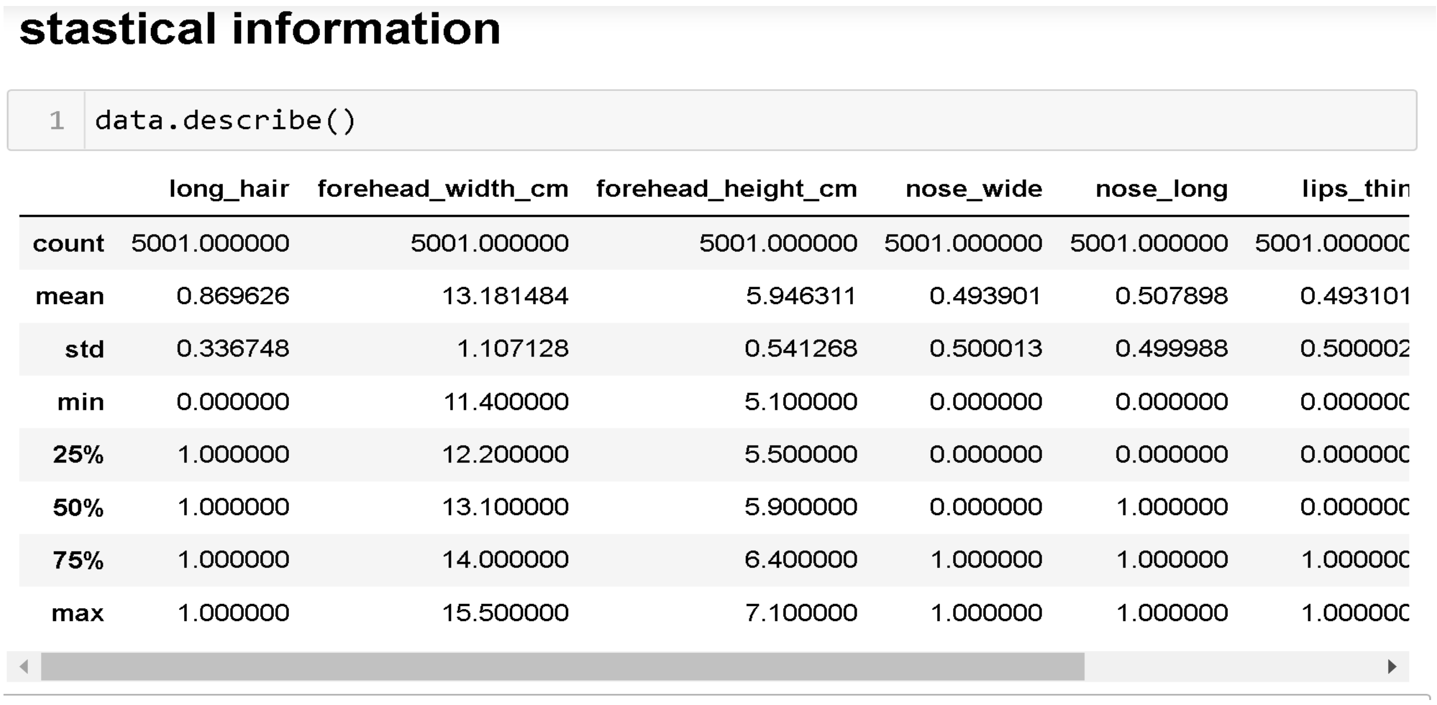
```
(5001, 8)
```

5001 rows and 8 columns

Information About Data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5001 entries, 0 to 5000
Data columns (total 8 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   long_hair                  5001 non-null   int64  
 1   forehead_width_cm          5001 non-null   float64
 2   forehead_height_cm         5001 non-null   float64
 3   nose_wide                  5001 non-null   int64  
 4   nose_long                  5001 non-null   int64  
 5   lips_thin                  5001 non-null   int64  
 6   distance_nose_to_lip_long  5001 non-null   int64  
 7   gender                     5001 non-null   object 
dtypes: float64(2), int64(5), object(1)
memory usage: 312.7+ KB
```



stastical information

```
data.describe()
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin
count	5001.000000	5001.000000	5001.000000	5001.000000	5001.000000	5001.000000
mean	0.869626	13.181484	5.946311	0.493901	0.507898	0.493101
std	0.336748	1.107128	0.541268	0.500013	0.499988	0.500002
min	0.000000	11.400000	5.100000	0.000000	0.000000	0.000000
25%	1.000000	12.200000	5.500000	0.000000	0.000000	0.000000
50%	1.000000	13.100000	5.900000	0.000000	1.000000	0.000000
75%	1.000000	14.000000	6.400000	1.000000	1.000000	1.000000
max	1.000000	15.500000	7.100000	1.000000	1.000000	1.000000

The describe() method returns a description of the data in the DataFrame.

If the DataFrame contains numerical data, the descript ion contains this information for each column:

count-The number of not-empty values.

mean - The average (mean) value.

max- maximum value of each feature

min - minimum value in the fw=eatures

std - The standard deviation.

25% - The 25% percentile*.

50% - The 50% percentile*.

75% - The 75% percentile*.

Percentile meaning: how many of the values are less t han the given percentile.

Checking Datatypes

```
data.dtypes
```

```
long_hair                      int64
forehead_width_cm            float64
forehead_height_cm           float64
nose_wide                      int64
nose_long                      int64
lips_thin                      int64
distance_nose_to_lip_long      int64
gender                        object
dtype: object
```

Checking null values

```
data.isnull().sum()
```

```
long_hair                    0
forehead_width_cm            0
forehead_height_cm           0
nose_wide                    0
nose_long                    0
lips_thin                    0
distance_nose_to_lip_long    0
gender                       0
dtype: int64
```

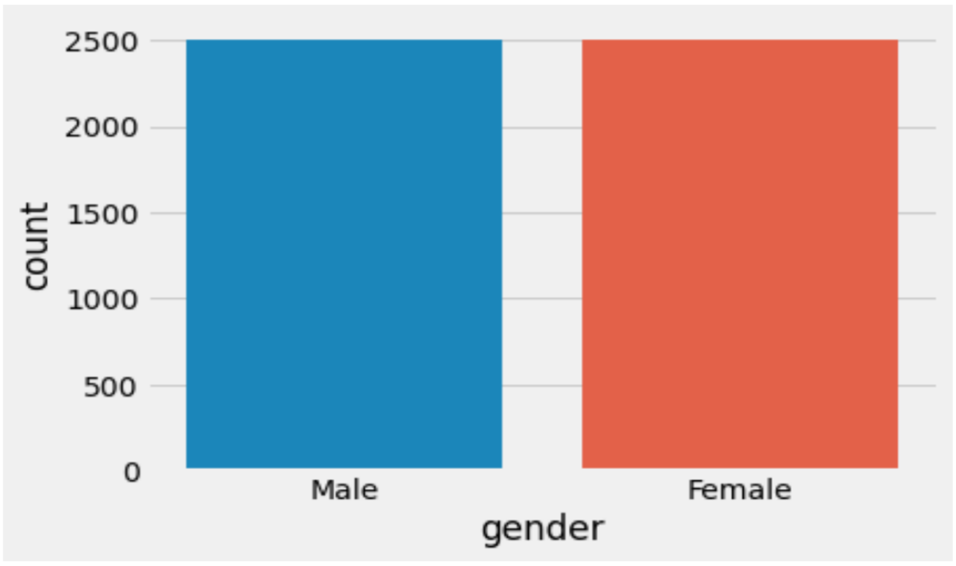
```
data.isna().sum().sum()
```

```
0
```

EDA-

Frequency of gender

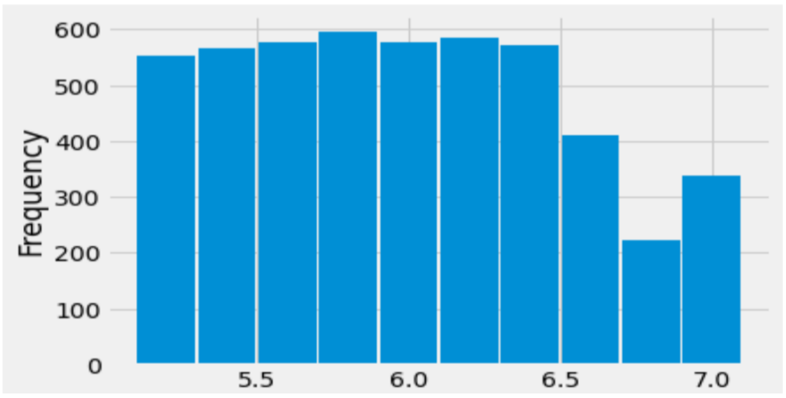
```
plt.style.use('fivethirtyeight')
sns.countplot(x=data['gender'])
plt.show()
```

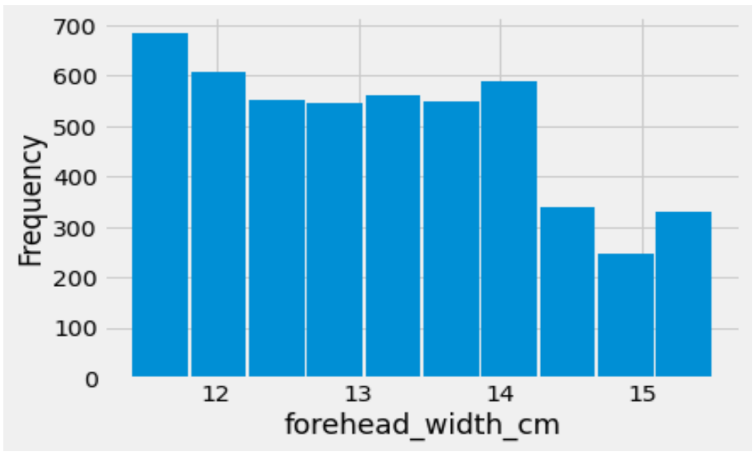


Observation- Both Gender has Equal Frequency

Frequency of forehead height and forehead width .

]:
```
cols2 = ['forehead_height_cm','forehead_width_cm']
for col in cols2:
    plt.style.use('fivethirtyeight')
    data[col].plot(kind='hist', rwidth=0.95)
    plt.xlabel(col)
    plt.show()
    print('\n')
```



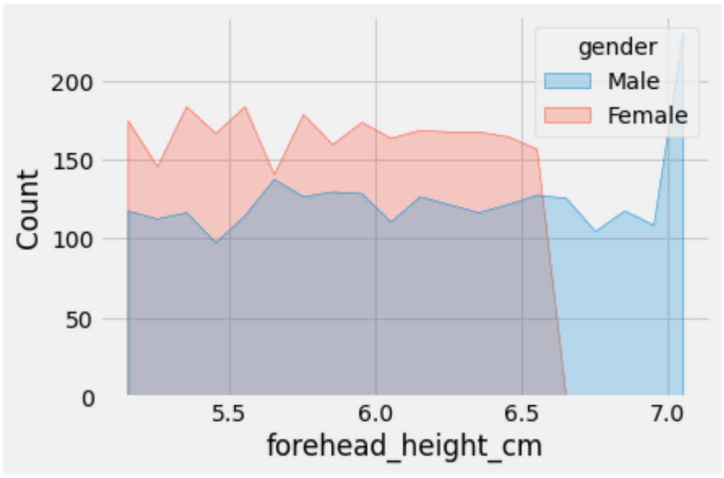


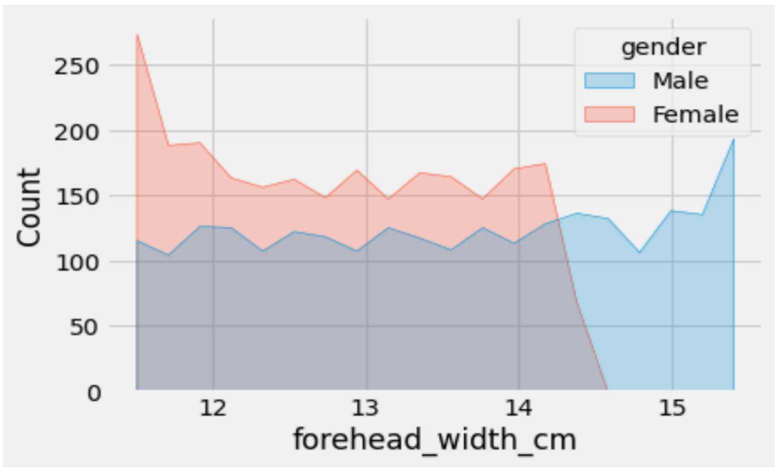
Observation- forehead height in cm of beween 5.5-6.0 has more frequency.

Observation- forehead width in cm of beween 0-12 has more frequency.

Count of forehead height and forehead width according to gender.

```
for col in cols2:
    sns.histplot(data=data[[col,'gender']],x=col, hue='gender',element='poly')
    plt.show()
```



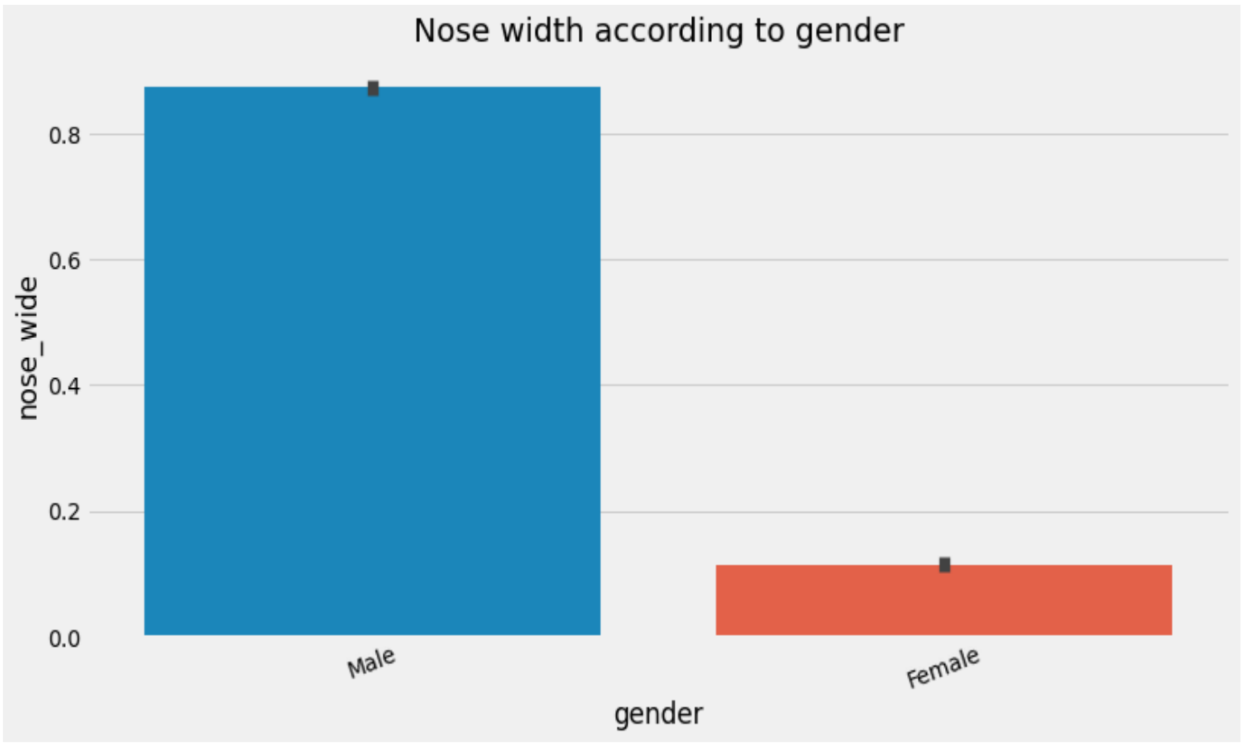


Observation- Forehead Height in cm Compared with Male and Female, Male has More Count than Female.

Observation- Forehead Width in cm Compared with Male and Female, Female has More Count than Male.

Nose width according to gender

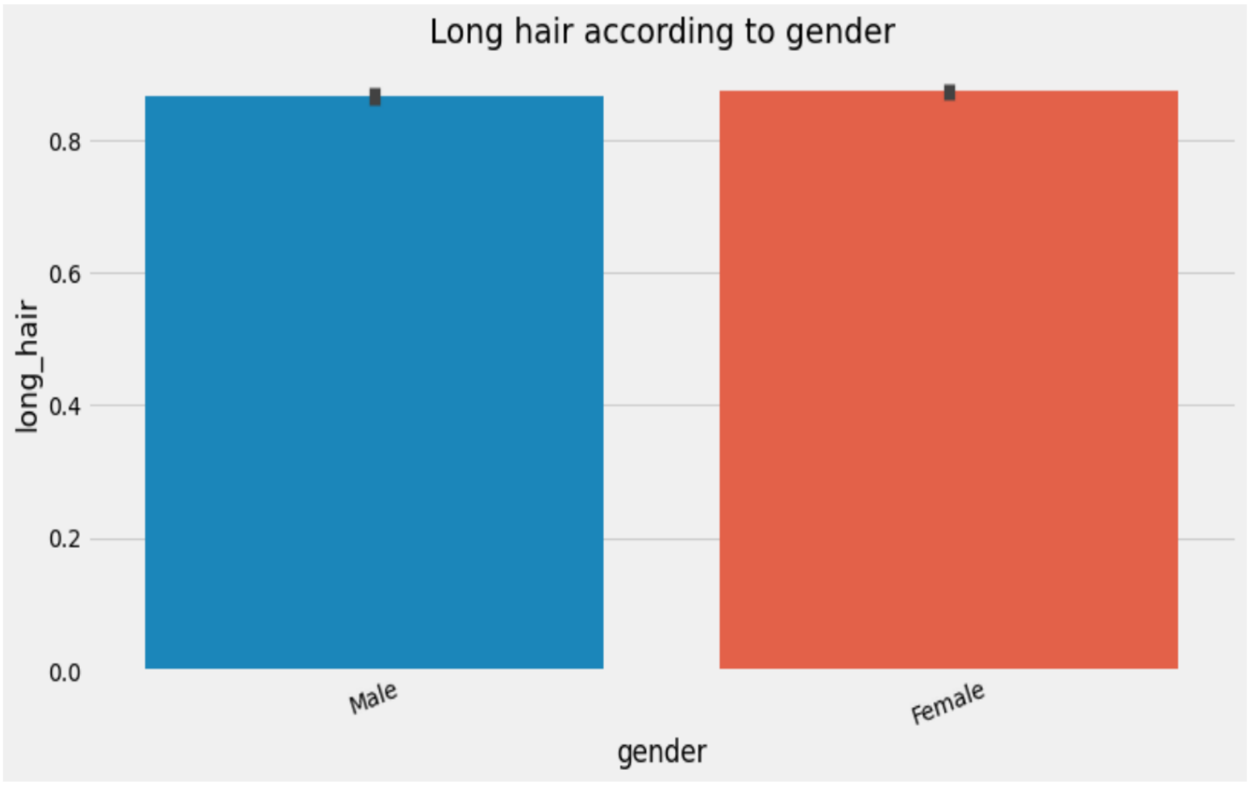
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Nose width according to gender')
sns.barplot(x=data['gender'],y=data['nose_wide'])
plt.show()
```



Observation- Nose Width in cm Compared with Male and Female, Male has More Nose Width than Female.

Long hair according to gender

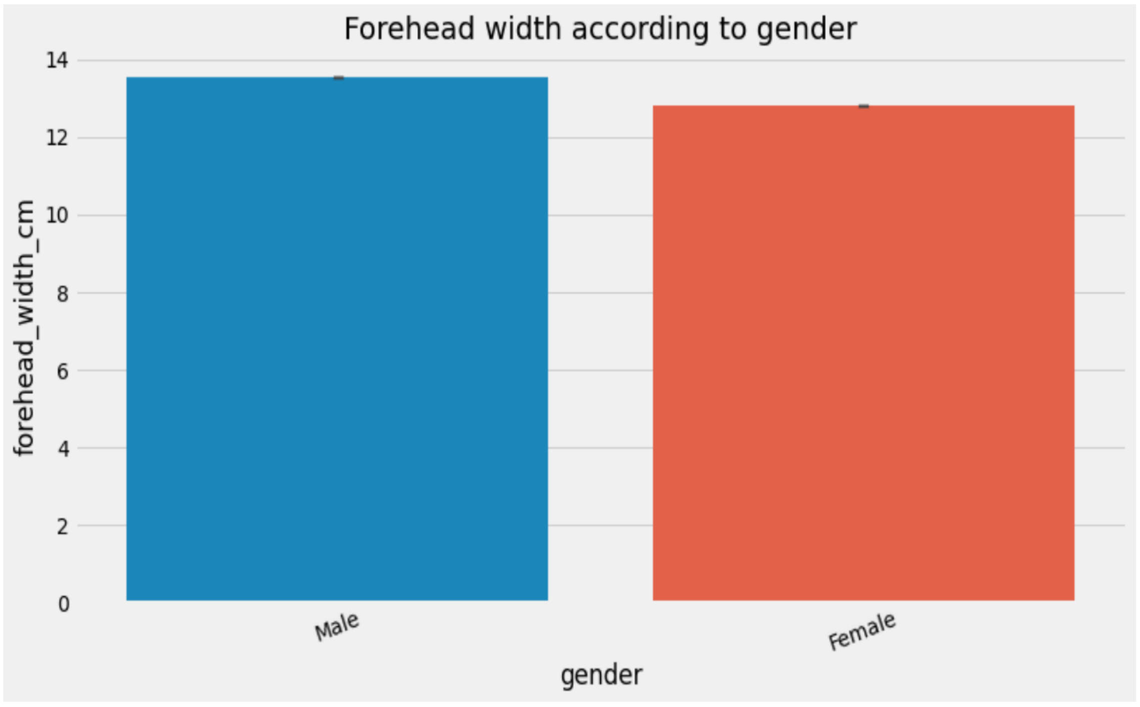
]:
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Long hair according to gender')
sns.barplot(x=data['gender'],y=data['long_hair'])
plt.show()
```



Observation- Long Hair Compared with Male and Female, Female has More Long hair than Male.

Forehead width according to gender

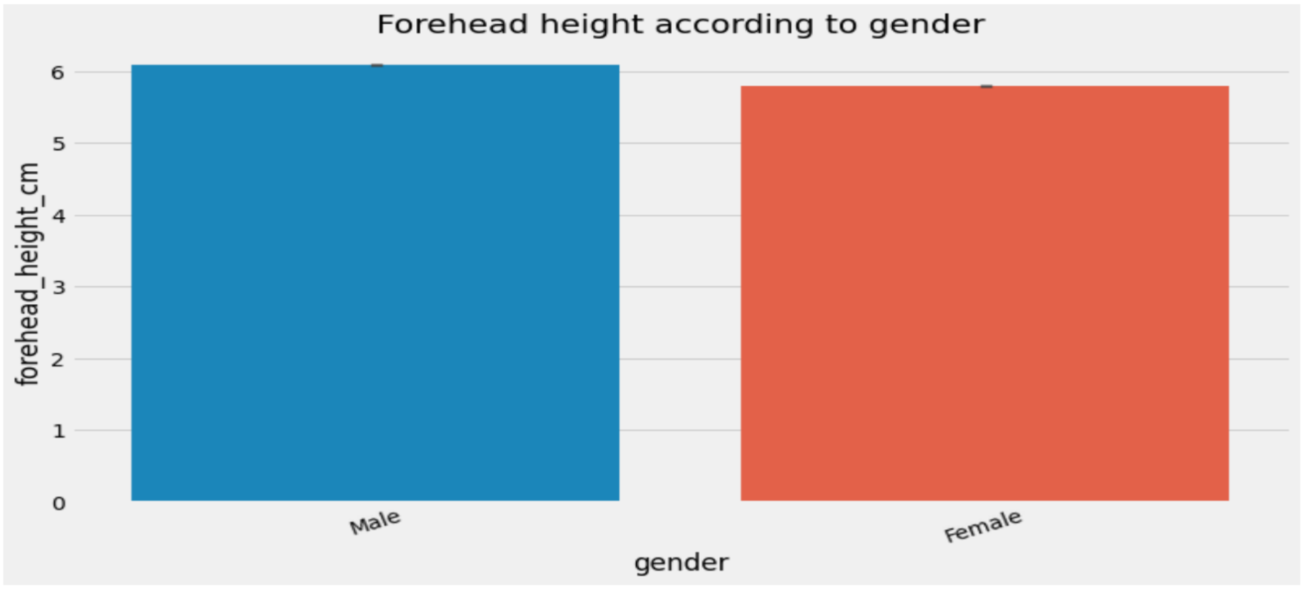
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Forehead width according to gender')
sns.barplot(x=data['gender'],y=data['forehead_width_cm'])
plt.show()
```



Observation- Forehead Width in cm Compared with Male and Female, Male has More Forehead Width than Female.

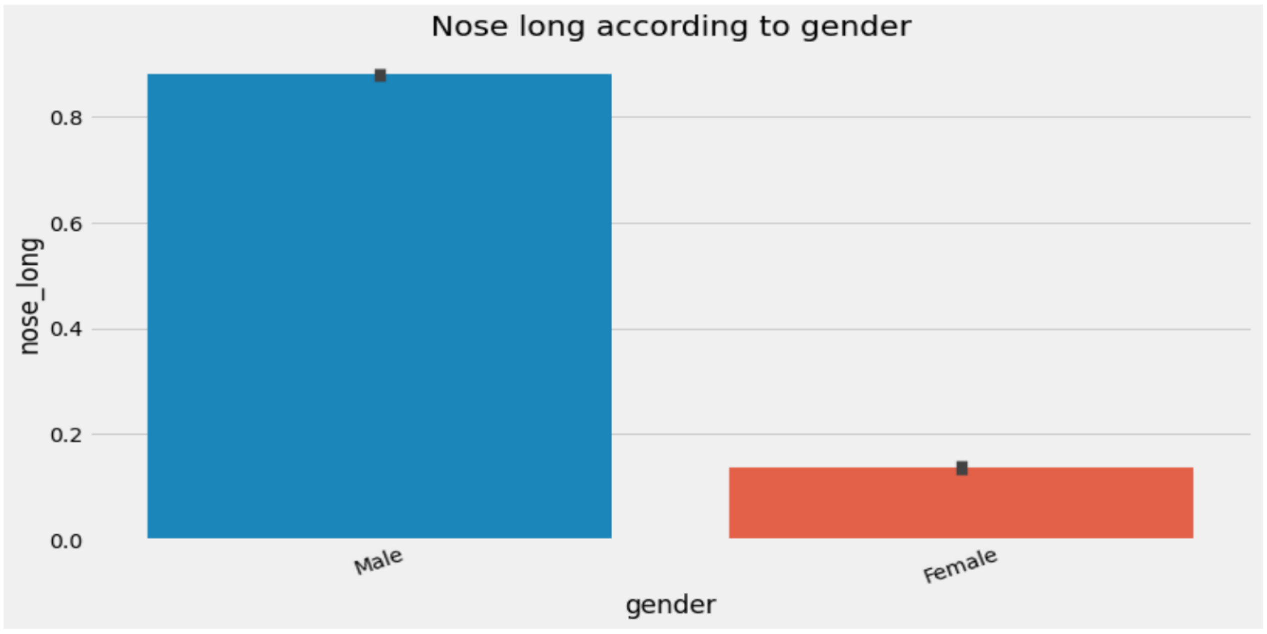
Forehead height according to gender

```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Forehead height according to gender')
sns.barplot(x=data['gender'],y=data['forehead_height_cm'])
plt.show()
```



Nose long according to gender

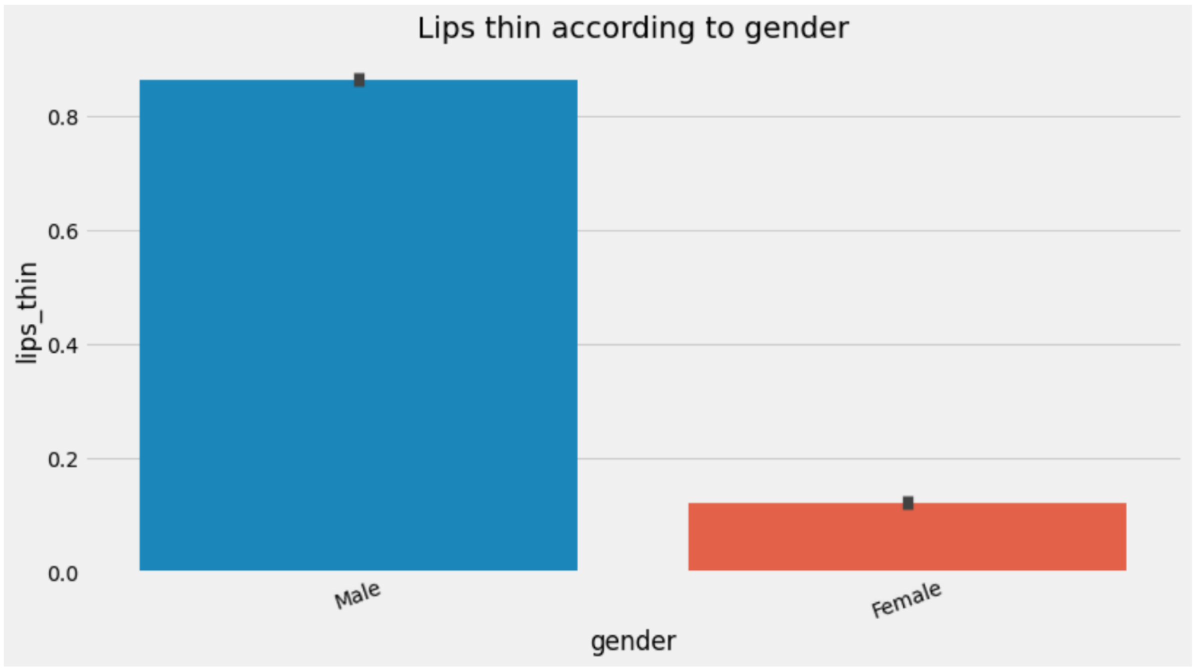
```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Nose long according to gender')
sns.barplot(x=data['gender'],y=data['nose_long'])
plt.show()
```



Observation-Nose Long Compared with Male and Female, Male has More Long Nose than Female.

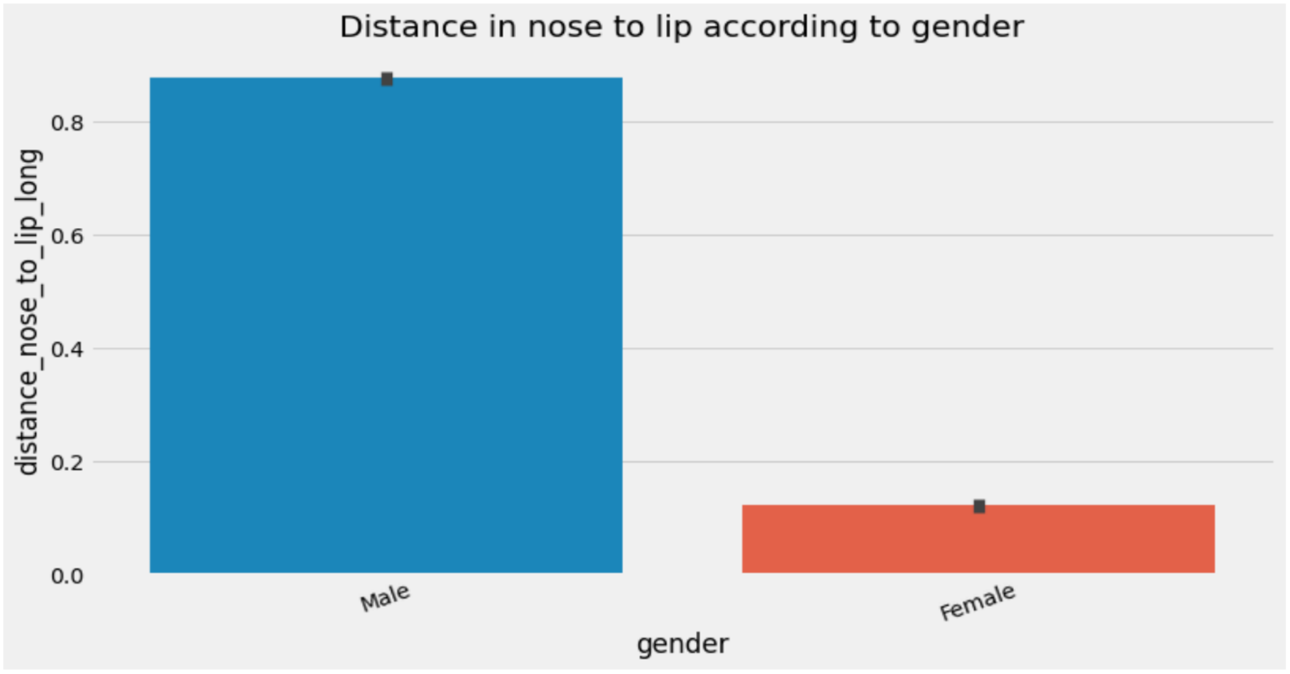
Lips thin according to gender

```
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Lips thin according to gender')
sns.barplot(x=data['gender'],y=data['lips_thin'])
plt.show()
```



Distance in nose to lip according to gender

```python
plt.figure(figsize=(12,6))
plt.xticks(rotation=20)
plt.title('Distance in nose to lip according to gender')
sns.barplot(x=data['gender'],y=data['distance_nose_to_lip_long'])
plt.show()
```



Observation-Distance in Nose to Lip Compared with Male and Female, Male has More Distance in Nose to Lip than Female.

Splitting Dataset

```python
x=data.drop(['gender'],axis=1)
y=data.gender
```

```python
x
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long
0	1	11.8	6.1	1	0	1	1
1	0	14.0	5.4	0	0	1	0
2	0	11.8	6.3	1	1	1	1
3	0	14.4	6.1	0	1	1	1
4	1	13.5	5.9	0	0	0	0
...	...	...	...	...	...	...	...
4996	1	13.6	5.1	0	0	0	0
4997	1	11.9	5.4	0	0	0	0
4998	1	12.9	5.7	0	0	0	0
4999	1	13.2	6.2	0	0	0	0
5000	1	15.4	5.4	1	1	1	1

5001 rows × 7 columns

```
y
```

```
0         Male
1       Female
2         Male
3         Male
4       Female
         ...  
4996    Female
4997    Female
4998    Female
4999    Female
5000      Male
Name: gender, Length: 5001, dtype: object
```

```
y.value_counts()
```

```
Female    2501
Male      2500
Name: gender, dtype: int64
```

Importing train-test

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4000, 7)
(1001, 7)
(4000,)
(1001,)
```

1-Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
data
```

	long_hair	forehead_width_cm	forehead_height_cm	nose_wide	nose_long	lips_thin	distance_nose_to_lip_long	gender
0	1	11.8	6.1	1	0	1	1	Male
1	0	14.0	5.4	0	0	1	0	Female
2	0	11.8	6.3	1	1	1	1	Male
3	0	14.4	6.1	0	1	1	1	Male
4	1	13.5	5.9	0	0	0	0	Female
...	...	...	...	...	...	...	...	...
4996	1	13.6	5.1	0	0	0	0	Female
4997	1	11.9	5.4	0	0	0	0	Female
4998	1	12.9	5.7	0	0	0	0	Female
4999	1	13.2	6.2	0	0	0	0	Female
5000	1	15.4	5.4	1	1	1	1	Male

5001 rows × 8 columns

```
model=LogisticRegression()
```

```
model.fit(x_train,y_train)
```

```
LogisticRegression()
```

```python
model.score(x_test,y_test)*100
```

```
96.7032967032967
```

```python
model.score(x_train,y_train)*100
```

```
97.15
```

Accuracy- 96.7%

2-Decision Tree

```python
from sklearn.tree import DecisionTreeClassifier
```

```python
model_dt=DecisionTreeClassifier()
```

```python
model_dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```python
model_dt.score(x_train,y_train)*100
```

```
99.875
```

```python
model_dt.score(x_test,y_test)*100
```

```
96.7032967032967
```

Accuracy- 96.7%

3-Random Forest

```python
from sklearn.ensemble import RandomForestClassifier
```

```python
model_rf=RandomForestClassifier(n_estimators=100)
```

```python
model_rf.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

```python
model_rf.score(x_train,y_train)*100
```

```
99.875
```

```python
model_rf.score(x_test,y_test)*100
```

```
97.30269730269731
```

Accuracy- 97%

4-KNeighborsClassifier

```python
from sklearn.neighbors import KNeighborsClassifier
```

```python
model_knn=KNeighborsClassifier(n_neighbors=10)
```

```python
model_knn.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=10)
```

```python
model_knn.score(x_train,y_train)*100
```

```
97.35000000000001
```

```python
model_knn.score(x_test,y_test)*100
```

```
96.90309690309691
```

Accuracy- 96.9%

5- Naive bayes

```
from sklearn.naive_bayes import MultinomialNB
```

```
model_nb=MultinomialNB()
```

```
model_nb.fit(x_train,y_train)
```

```
MultinomialNB()
```

```
model_nb.score(x_train,y_train)*100
```

```
95.8
```

```
model_nb.score(x_test,y_test)*100
```

```
95.7042957042957
```

Accuracy- 95.7%

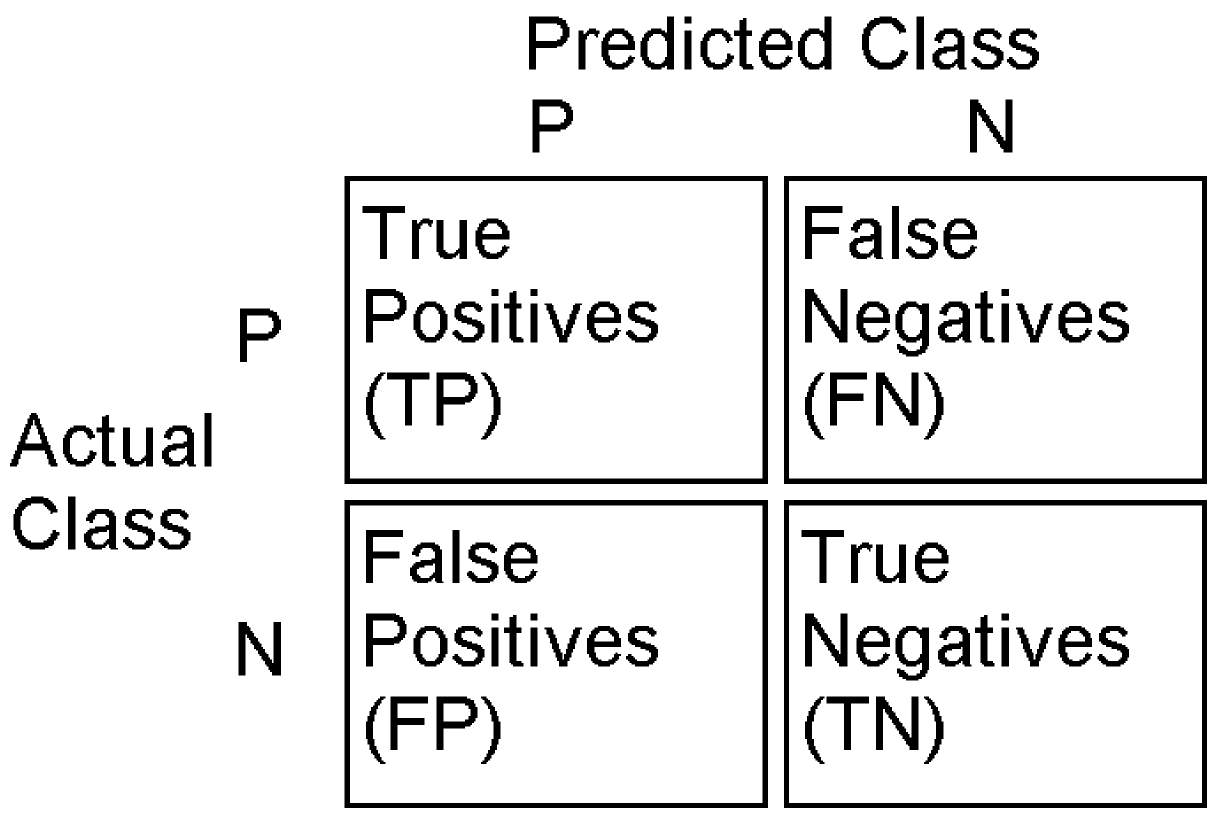
So, here the Accuracies’ are as follows:-

1. Logistic Regression:- 96.7%
2. Decision Tree:- 96.7%
3. Random Forest:- 97.3%
4. KNeighboursClassifier:- 96.9%
5. Naïve Bayes:- 95.7%

So, as Random Forest has the Highest Accuracy amongst all Algorithms therefore we use Random Forest for Predicting.

Measuring Model Performance:-

While there are other ways of measuring model performance (precision, recall, F1 Score, ROC Curve, etc), let's keep this simple and use accuracy as our metric. To do this are going to see how the model performs on new data (test set) Accuracy is defined as (a fraction of correct predictions): correct predictions / total number of data points



```python
from sklearn.metrics import confusion_matrix
```

```python
performance=confusion_matrix(y_test,y_predict)
```

```python
performance
```

```
array([[502,  13],
       [ 14, 472]], dtype=int64)
```

Here, 502 is the number of True Positives in our data, while 472 is the number of True Negatives. 13 & 14 are the number of errors. There are 13 type-1 error (False Positives)- You predicted positive and it’s false. There are 14 type-2 error (False Negatives)- You predicted negative and it’s false.

Hence, if we calculate the accuracy it’s #Correct Predicted/ # Total. In other words, where TP, FN, FP, and TN represent the number of true positives, false negatives, false positives, and true negatives. (TP + TN)/(TP + TN + FP + FN). (502 +472)/(502 +472 +13 +14) = 0.80 = 97.3% accuracy.

Note: A good rule of thumb is that any accuracy above 70% is considered good, but be careful because if your accuracy is extremely high, it may be too good to be true (an example of Overfitting). Thus, 97.3% is the ideal accuracy!

Predicting:-

```
y_predict=model_rf.predict(x_test)
```

```
y_predict
```

```
array(['Female', 'Female', 'Female', ..., 'Female', 'Male', 'Male'],
      dtype=object)
```

```
data_1=pd.DataFrame({'Actual':y_test,'Predicted':y_predict})
```

```
data_1
```

```
data_1=pd.DataFrame({'Actual':y_test,'Predicted':y_predict})
```

```
data_1
```

	Actual	Predicted
3131	Female	Female
4465	Female	Female
4139	Female	Female
4855	Male	Male
2669	Male	Male
...	...	...
2914	Female	Female
3318	Male	Male
4746	Female	Female
4367	Male	Male
1297	Male	Male

1001 rows × 2 columns

```
from sklearn.metrics import classification_report
```

```
performance=classification_report(y_test,y_predict)
```

```
print(performance)
```

```
              precision    recall  f1-score   support

      Female       0.97      0.97      0.97       515
        Male       0.97      0.97      0.97       486

    accuracy                           0.97      1001
   macro avg       0.97      0.97      0.97      1001
weighted avg       0.97      0.97      0.97      1001
```

Conclusions:-

Our Random Forest algorithm yields the highest accuracy, 97%. Any accuracy above 70% is considered good.

We can Identify Gender either Male or Female by the given Features.