In [12]:

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

In [ ]:

```

```

In [14]:

```
!pip install opencv-python
import cv2
```

```
Requirement already satisfied: opencv-python in c:\users\shahzad\anaconda3
\new folder (2)\lib\site-packages (4.8.0.74)
Requirement already satisfied: numpy>=1.17.3 in c:\users\shahzad\anaconda3
\new folder (2)\lib\site-packages (from opencv-python) (1.23.5)
```

In [15]:

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

In [16]:

```
dataset=[]
```

FOLDER PATH

In [17]:

```
folder_path=['C:/Users/shahzad/Downloads/gender/train/female','C:/Users/shahzad/Downloads
```

In [18]:

```
import os
```

Data Preprocessing

In [30]:

```
# Iterate over the folder paths
for i in folder_path:
    folder_name = os.path.basename(i)

 # Iterate over the images in the subdirectory
    for file_name in os.listdir(i):
        image_path = os.path.join(i, file_name)
        if os.path.isfile(image_path): # Only consider files
            # Load the image using OpenCV
            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
            # If the image was successfully loaded
            if image is not None:
                # Resize the grayscale image to 100X100 pixels
                resized_image = cv2.resize(image, (100, 100))
                # Flatten the image and append each pixel as a separate feature along wit
                flattened_image = resized_image.flatten().tolist()
                label = 'male' if 'male' in file_name else 'female'  # Adjust label assig
            dataset.append(flattened_image + [label])
        else:
            print(f"Error loading image: {image_path}")

image_size = 100
num_pixels = image_size * image_size
column_names = [f'pixel_{i+1}' for i in range(num_pixels)] + ['label']
df = pd.DataFrame(dataset, columns=column_names)

print(df.head())
```

```
   pixel_1  pixel_2  pixel_3  pixel_4  pixel_5  pixel_6  pixel_7  pixel_8
\
0       42       37       32       32       35       29       22       17
1       53       57       55       51       58       64       64       64
2       12       11        9        8        9       13       15       17
3      140      148      150      159      178      191      188      174
4       32       20       12       11        1        8       18       20

   pixel_9  pixel_10  ...  pixel_9992  pixel_9993  pixel_9994  pixel_9995
\
0       17        20  ...           2           1           0           1
1       68        67  ...         109         109         108         106
2       19        20  ...          19          23          30          46
3      166       172  ...          39          28          25          25
4       17        26  ...         108         103         112         108

   pixel_9996  pixel_9997  pixel_9998  pixel_9999  pixel_10000  label
0           7          15          21          21           17   male
1         103          99          94          89           85   male
2          59          48          47          48           41   male
3          29          36          39          38           35   male
4          90          94         104          96          101   male

[5 rows x 10001 columns]
```

In [ ]:

```
```

In [20]:

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['label_encoded'] = label_encoder.fit_transform(df['label'])
```

In [31]:

```
label = 'male' if 'male' in file_name.lower() else 'female'
```

In [32]:

```
unique_labels = df['label'].unique()
print(unique_labels)
```

```
['male' 'female']
```

DATASET SPLITTING(train,test)

In [34]:

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns=['label']),df['label'

# Print the shapes of the resulting sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (5585, 10000)
X_test shape: (1397, 10000)
y_train shape: (5585,)
y_test shape: (1397,)
```

MACINE LEARNING MODELS

SVM

In [35]:

```
label_counts = df['label'].value_counts()
print(label_counts)
```

```
male      3491
female    3491
Name: label, dtype: int64
```

In [ ]:

```

```

In [ ]:

```

```

In [25]:

```
from sklearn.metrics import confusion_matrix,classification_report

```

In [26]:

```
from sklearn.svm import SVC
```

In [27]:

```
model_svc=SVC()
```

In [37]:

```
df.rename(columns={df.iloc[:,-1].name:'Target'},inplace=True)
```

In [38]:

```
#get num of rows of dataset
num_rows=len(df)
#generate permutated indices
permuted_indices=np.random.permutation(num_rows)
#generate random data
random_df=df.iloc[permuted_indices]
```

In [39]:

```
X=random_df.drop('Target',axis=1)
X=X/255
X.head()
```

Out[39]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_
5601	0.113725	0.109804	0.113725	0.125490	0.137255	0.149020	0.121569	0.113725	0.1294
2108	0.113725	0.145098	0.168627	0.168627	0.156863	0.152941	0.180392	0.243137	0.3490
2516	0.082353	0.074510	0.070588	0.082353	0.129412	0.207843	0.301961	0.376471	0.4274
173	0.050980	0.054902	0.062745	0.066667	0.066667	0.066667	0.062745	0.062745	0.0627
33	0.011765	0.035294	0.039216	0.043137	0.082353	0.184314	0.227451	0.172549	0.1137

5 rows × 10000 columns

In [41]:

```
encoder=LabelEncoder()
y=random_df.Target
y_encoded=encoder.fit_transform(y)
y_series=pd.Series(y_encoded,name='target')
```

In [42]:

```
from sklearn.model_selection import train_test_split
np.random.seed(42)
X_train,X_test,y_train,y_test=train_test_split(X,y_encoded,test_size=0.2,random_state=42)
```

In [43]:

```
model_svc.fit(X_train,y_train);
```

In [44]:

```
y_pred_svc=model_svc.predict(X_test)
```

In [46]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [47]:

```
Accuracy_svc=accuracy_score(y_pred_svc,y_test)
print('Accuracy:',Accuracy_svc)
CR=classification_report(y_pred_svc,y_test)
print('Classification Report\n',CR)
cm=confusion_matrix(y_pred_svc,y_test)
```

```
Accuracy: 0.28418038654259126
Classification Report
               precision    recall  f1-score   support

           0       0.10      0.18      0.13       414
           1       0.49      0.33      0.39       983

    accuracy                           0.28      1397
   macro avg       0.29      0.25      0.26      1397
weighted avg       0.37      0.28      0.31      1397
```

In [54]:

```
cm
```

Out[54]:

```
array([[153, 520],
       [579, 145]], dtype=int64)
```

Deployment

In [69]:

```
import matplotlib.pyplot as plt
image_path = "C:/Users/shahzad/Downloads/gender_rev2/test/male/024"
user_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Resize the image to match the input size expected by the model
resized_image = cv2.resize(user_image, (100, 100))

# Flatten the image
flattened_img = resized_image.flatten()

# Normalize the flattened image data
normalized_user_image = flattened_img / 255.0

# Convert the normalized flattened image to a NumPy array and reshape it
user_input = normalized_user_image.reshape(1, -1)

# Make a prediction using the trained model
user_prediction = model_svc.predict(user_input)
image=cv2.cvtColor(resized_image,cv.COLOR_BGR2RGB)
# Decode the predicted label
predicted_class = encoder.inverse_transform(user_prediction)[0]
plt.imshow(image)
plt.title(predicted_class)
```

```
---------------------------------------------------------------------------
-
error                                     Traceback (most recent call las
t)
Cell In[69], line 6
      3 user_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
      5 # Resize the image to match the input size expected by the model
----> 6 resized_image = cv2.resize(user_image, (100, 100))
      8 # Flatten the image
      9 flattened_img = resized_image.flatten()

error: OpenCV(4.8.0) D:\a\opencv-python\opencv-python\opencv\modules\imgpr
oc\src\resize.cpp:4062: error: (-215:Assertion failed) !ssize.empty() in f
unction 'cv::resize'
```

Decision Tree

In [ ]:

```

```

In [26]:

```
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the LFW (Labeled Faces in the Wild) dataset
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

X = lfw_people.images
y = lfw_people.target_names[lfw_people.target]

# Flatten the images into 1D arrays
X = X.reshape(X.shape[0], -1)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a decision tree classifier
clf = DecisionTreeClassifier()

# Train the classifier
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

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

```
Accuracy: 0.47674418604651164
```

logistic regression

In [49]:

```
from sklearn.linear_model import LogisticRegression
model_log=LogisticRegression()
```

In [ ]:

```

```

In [50]:

```
model_log.fit(X_train,y_train);
```

```
C:\Users\shahzad\anaconda3\New folder (2)\lib\site-packages\sklearn\linear
_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (sta
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

In [51]:

```
y_pred_log=model_log.predict(X_test)
```

In [52]:

```
accuracy_log=accuracy_score(y_pred_log,y_test)
print("Accuracy Score is :",accuracy_log)
C_report=classification_report(y_pred_log,y_test)
print('Classification report:',C_report)
cm=confusion_matrix(y_pred_log,y_test)
```

```
Accuracy Score is : 0.2133142448103078
Classification report:               precision    recall  f1-score   suppo
rt

           0       0.21      0.23      0.22       673
           1       0.22      0.20      0.21       724

    accuracy                           0.21      1397
   macro avg       0.21      0.21      0.21      1397
weighted avg       0.21      0.21      0.21      1397
```

In [55]:

```
cm
```

Out[55]:

```
array([[153, 520],
       [579, 145]], dtype=int64)
```

In [ ]:

```

```

KNN

In [58]:

```
from sklearn.neighbors import KNeighborsClassifier
model_KNN=KNeighborsClassifier()
model_KNN.fit(X_train,y_train)
```

Out[58]:

```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [59]:

```
y_pred_knn=model_KNN.predict(np.array(X_test))
```

```
C:\Users\shahzad\anaconda3\New folder (2)\lib\site-packages\sklearn\base.p
y:420: UserWarning: X does not have valid feature names, but KNeighborsCla
ssifier was fitted with feature names
  warnings.warn(
```

In [60]:

```
accuracy_knn=accuracy_score(y_pred_knn,y_test)
print("Accuracy Score is :",accuracy_knn)
C_report=classification_report(y_pred_knn,y_test)
print('Classification report:',C_report)
cm=confusion_matrix(y_pred_knn,y_test)
```

```
Accuracy Score is : 0.2297780959198282
Classification report:               precision    recall  f1-score   suppo
rt

           0       0.21      0.23      0.22       646
           1       0.26      0.23      0.24       751

    accuracy                           0.23      1397
   macro avg       0.23      0.23      0.23      1397
weighted avg       0.23      0.23      0.23      1397
```

In [62]:

```
cm
```

Out[62]:

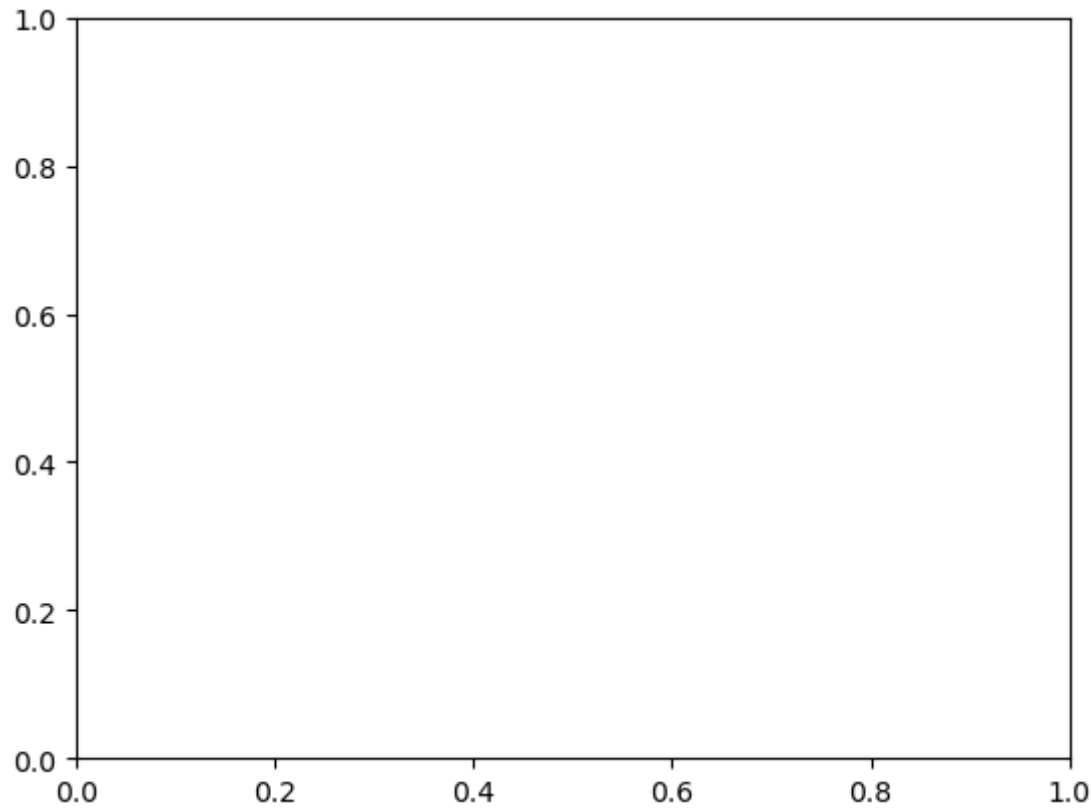
```
array([[151, 495],
       [581, 170]], dtype=int64)
```

In [65]:

```
import matplotlib.pyplot as plt

algorithms = ['SVM', 'KNN', 'Decision Tree', 'Logistic Regression']
accuracy = [Accuracy_svc, accuracy_knn,accuracy_log]

colors = ['red', 'pink', 'blue', 'black']
bar_width = 0.5  # Width of the bars
bar_positions = range(len(algorithms))

plt.bar(bar_positions, accuracy, color=colors, width=bar_width)
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison of Machine Learning Algorithms')
plt.ylim(0, 1)  # Set y-axis limits between 0 and 1

# Annotate each bar with its accuracy value
for i, acc in enumerate(accuracy):
    plt.text(i, acc + 0.02, f'{acc:.2f}', ha='center')

# Adjust x-axis labels and positions
plt.xticks(bar_positions, algorithms, rotation=15, ha='right')

plt.tight_layout()  # To prevent labels from being cut off
plt.show()
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
Cell In[65], line 10
      7 bar_width = 0.5  # Width of the bars
      8 bar_positions = range(len(algorithms))
---> 10 plt.bar(bar_positions, accuracy, color=colors, width=bar_width)
     11 plt.xlabel('Algorithms')
     12 plt.ylabel('Accuracy')

File ~\anaconda3\New folder (2)\lib\site-packages\matplotlib\pyplot.py:241
2, in bar(x, height, width, bottom, align, data, **kwargs)
   2408 @_copy_docstring_and_deprecators(Axes.bar)
   2409 def bar(
   2410         x, height, width=0.8, bottom=None, *, align='center',
   2411         data=None, **kwargs):
-> 2412     return gca().bar(
   2413         x, height, width=width, bottom=bottom, align=align,
   2414         **({"data": data} if data is not None else {}), **kwargs)

File ~\anaconda3\New folder (2)\lib\site-packages\matplotlib\__init__.py:1
442, in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1439 @functools.wraps(func)
   1440 def inner(ax, *args, data=None, **kwargs):
   1441     if data is None:
-> 1442         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1444     bound = new_sig.bind(ax, *args, **kwargs)
   1445     auto_label = (bound.arguments.get(label_namer)
   1446                   or bound.kwargs.get(label_namer))

File ~\anaconda3\New folder (2)\lib\site-packages\matplotlib\axes\_axes.p
y:2417, in Axes.bar(self, x, height, width, bottom, align, **kwargs)
   2414     if yerr is not None:
   2415         yerr = self._convert_dx(yerr, y0, y, self.convert_yunits)
-> 2417 x, height, width, y, linewidth, hatch = np.broadcast_arrays(
   2418     # Make args iterable too.
   2419     np.atleast_1d(x), height, width, y, linewidth, hatch)
   2421 # Now that units have been converted, set the tick locations.
   2422 if orientation == 'vertical':

File <__array_function__ internals>:180, in broadcast_arrays(*args, **kwar
gs)

File ~\anaconda3\New folder (2)\lib\site-packages\numpy\lib\stride_tricks.
py:540, in broadcast_arrays(subok, *args)
    533 # nditer is not used here to avoid the limit of 32 arrays.
    534 # Otherwise, something like the following one-liner would suffice:
    535 # return np.nditer(args, flags=['multi_index', 'zerosize_ok'],
    536 #                  order='C').itviews
    538 args = [np.array(_m, copy=False, subok=subok) for _m in args]
--> 540 shape = _broadcast_shape(*args)
    542 if all(array.shape == shape for array in args):
    543     # Common case where nothing needs to be broadcasted.
    544     return args

File ~\anaconda3\New folder (2)\lib\site-packages\numpy\lib\stride_tricks.
py:422, in _broadcast_shape(*args)
    417 """Returns the shape of the arrays that would result from broadcas
ting the
    418 supplied arrays against each other.
```

```
    419 """
    420 # use the old-iterator because np.nditer does not handle size 0 ar
rays
    421 # consistently
--> 422 b = np.broadcast(*args[:32])
    423 # unfortunately, it cannot handle 32 or more arguments directly
    424 for pos in range(32, len(args), 31):
    425     # ironically, np.broadcast does not properly handle np.broadca
st
    426     # objects (it treats them as scalars)
    427     # use broadcasting to avoid allocating the full array

ValueError: shape mismatch: objects cannot be broadcast to a single shape.
Mismatch is between arg 0 with shape (4,) and arg 1 with shape (3,).
```



In [ ]:

In [ ]: