```
import pandas as pd
from sklearn.impute import KNNImputer
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix


!wget https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/X_Test_Data_Input.csv
!wget https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/X_Train_Data_Input.csv
!wget https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/Y_Test_Data_Target.csv
!wget https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/Y_Train_Data_Target.csv
```

```
--2024-09-08 06:39:19--  https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/X_Test_Data_Input.csv
Resolving temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)... 52.219.158.146, 52.219.
Connecting to temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)|52.219.158.146|:443...
HTTP request sent, awaiting response... 200 OK
Length: 56005366 (53M) [text/csv]
Saving to: 'X_Test_Data_Input.csv'

X_Test_Data_Input.c 100%[===================>]  53.41M  12.2MB/s    in 5.0s

2024-09-08 06:39:25 (10.7 MB/s) - 'X_Test_Data_Input.csv' saved [56005366/56005366]

--2024-09-08 06:39:25--  https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/X_Train_Data_Input.csv
Resolving temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)... 52.219.158.202, 16.12.3
Connecting to temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)|52.219.158.202|:443...
HTTP request sent, awaiting response... 200 OK
Length: 168002518 (160M) [text/csv]
Saving to: 'X_Train_Data_Input.csv'

X_Train_Data_Input. 100%[===================>] 160.22M  12.6MB/s    in 15s

2024-09-08 06:39:41 (10.9 MB/s) - 'X_Train_Data_Input.csv' saved [168002518/168002518]

--2024-09-08 06:39:41--  https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/Y_Test_Data_Target.csv
Resolving temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)... 52.219.64.83, 16.12.36.
Connecting to temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)|52.219.64.83|:443... c
HTTP request sent, awaiting response... 200 OK
Length: 9159930 (8.7M) [text/csv]
Saving to: 'Y_Test_Data_Target.csv'

Y_Test_Data_Target. 100%[===================>]   8.74M  4.56MB/s    in 1.9s

2024-09-08 06:39:44 (4.56 MB/s) - 'Y_Test_Data_Target.csv' saved [9159930/9159930]

--2024-09-08 06:39:44--  https://temp-devesh.s3.ap-south-1.amazonaws.com/gst/Y_Train_Data_Target.csv
Resolving temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)... 52.219.64.83, 16.12.36.
Connecting to temp-devesh.s3.ap-south-1.amazonaws.com (temp-devesh.s3.ap-south-1.amazonaws.com)|52.219.64.83|:443... c
HTTP request sent, awaiting response... 200 OK
Length: 27479665 (26M) [text/csv]
Saving to: 'Y_Train_Data_Target.csv'

Y_Train_Data_Target 100%[===================>]  26.21M  8.07MB/s    in 3.2s

2024-09-08 06:39:48 (8.07 MB/s) - 'Y_Train_Data_Target.csv' saved [27479665/27479665]
```

```
# training dataset
df_train_x = pd.read_csv('X_Train_Data_Input.csv')
df_train_y = pd.read_csv('Y_Train_Data_Target.csv')

df_train = pd.concat([df_train_x, df_train_y], axis=1)
df_train = df_train.drop('ID', axis=1)

print(len(df_train))
df_train.head()
```

785133

	Column0	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Colu
0	2.0	2495	3726.0	0.678139	0.701403	-0.007468	0.434190	-0.015603	0.606
1	0.0	2495	3454.0	0.452580	0.701403	-0.007468	1.554998	-0.015574	0.329
2	2.0	2495	4543.0	-1.577453	-1.429540	-0.007469	-0.407939	-0.015607	-0.774
3	0.0	211	59.0	NaN	NaN	NaN	-0.407939	-0.015607	-0.774
4	0.0	718	950.0	-2.028572	-1.855728	NaN	-0.407939	-0.015607	-0.774

5 rows × 23 columns

```
def show_missing(df):

    for column in df.columns:
        missing_values = df[column].isnull().sum()
        total_rows = len(df)
        percentage_missing = (missing_values / total_rows) * 100
        if percentage_missing: print(f"{column} : {percentage_missing}%")

show_missing(df_train)
```

```
Column0 : 0.0011463026009606015%
Column3 : 16.086828601014098%
Column4 : 16.266033907630938%
Column5 : 21.293207647621486%
Column6 : 0.49036277929981287%
Column8 : 0.49036277929981287%
Column9 : 93.25006081772132%
Column14 : 46.5784777865661%
Column15 : 2.095950622378629%
```

```
def preprocess(df):
    # for each column in the dataframe 'df'
    # get the maximum and minimum of that column
    # for each datacell in that columns which does not have 'NaN'
    # normalise the data in it to be between 0 and 1

    for column in df.columns:
        min_val = df[column].min()
        max_val = df[column].max()
        df[column] = (df[column] - min_val) / (max_val - min_val)

    df['Column9'] = df['Column9'].notna().astype(int)
    df['Column14'] = df['Column14'].notna().astype(int)


    columns_to_impute = ['Column0']
    imputer = KNNImputer(n_neighbors=5)
    df[columns_to_impute] = imputer.fit_transform(df[columns_to_impute])

    columns_to_impute_mean = ['Column3', 'Column4', 'Column5', 'Column6', 'Column8', 'Column15']
    for column in columns_to_impute_mean:
        df[column] = df[column].fillna(df[column].mean())

    return df


df_train = preprocess(df_train)
df_train.head()
```

	Column0	Column1	Column2	Column3	Column4	Column5	Column6	Column7
0	0.111111	0.998882	0.365000	1.000000	1.000000	1.333866e-09	0.054706	2.300930e-08
1	0.000000	0.998882	0.338686	0.916667	1.000000	1.383023e-09	0.127515	1.665141e-07
2	0.111111	0.998882	0.444036	0.166667	0.166667	0.000000e+00	0.000000	0.000000e+00
3	0.000000	0.360715	0.010254	0.749382	0.725373	1.287787e-05	0.000000	0.000000e+00
4	0.000000	0.502375	0.096450	0.000000	0.000000	1.287787e-05	0.000000	0.000000e+00

5 rows × 23 columns

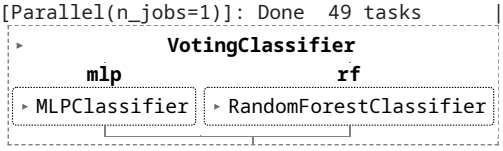
```
X = df_train.drop('target', axis=1)
y = df_train['target']

mlp = MLPClassifier(max_iter=15, verbose=1)
rf = RandomForestClassifier(n_estimators=100, verbose=1)
svc = SVC(probability=True, verbose=1)

# Create the ensemble model
ensemble_model = VotingClassifier(estimators=[
    ('mlp', mlp),
    ('rf', rf),
    # ('svc', svc)
], voting='soft')

# Train the ensemble model
ensemble_model.fit(X, y)
```

```
Iteration 1, loss = 0.10623832
Iteration 2, loss = 0.06537381
Iteration 3, loss = 0.06151325
Iteration 4, loss = 0.06094759
Iteration 5, loss = 0.06037571
Iteration 6, loss = 0.05985833
Iteration 7, loss = 0.05940790
Iteration 8, loss = 0.05894509
Iteration 9, loss = 0.05862103
Iteration 10, loss = 0.05832658
Iteration 11, loss = 0.05809414
Iteration 12, loss = 0.05790454
Iteration 13, loss = 0.05765799
Iteration 14, loss = 0.05753704
Iteration 15, loss = 0.05732542
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perce
  warnings.warn(
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:   28.4s
```




```
# testing dataset
df_test_x = pd.read_csv('X_Test_Data_Input.csv')
df_test_y = pd.read_csv('Y_Test_Data_Target.csv')

df_test = pd.concat([df_test_x, df_test_y], axis=1)
df_test = df_test.drop('ID', axis=1)

df_test = preprocess(df_test)
print(len(df_test))
```

```
261712
```

```
X_test = df_test.drop('target', axis=1)
y_test = df_test['target']

y_pred = ensemble_model.predict(X_test)
```

```
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:    1.2s
```

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

# Calculate precision
precision = precision_score(y_test, y_pred)
print("Precision:", precision)

# Calculate recall
recall = recall_score(y_test, y_pred)
print("Recall:", recall)

# Calculate F1 score
f1 = f1_score(y_test, y_pred)
print("F1 Score:", f1)

# Calculate AUC-ROC
auc_roc = roc_auc_score(y_test, y_pred)
print("AUC-ROC:", auc_roc)

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Accuracy: 0.9759162743779422
Precision: 0.8235376976441173
Recall: 0.9476456763108841
F1 Score: 0.8812435233160623
AUC-ROC: 0.9632526245995809
Confusion Matrix:
 [[232023   5011]
 [  1292  23386]]
```