```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.preprocessing import StandardScaler
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.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
```

```
from tabulate import tabulate
```

```
data = pd.read_csv('/content/Forest Cover Type.csv')
```

```
data.head()
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshad
0	2596	51	3	258	0	510	
1	2590	56	2	212	-6	390	
2	2804	139	9	268	65	3180	
3	2785	155	18	242	118	3090	
4	2595	45	2	153	-1	391	

5 rows × 55 columns

```
data.tail()
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hill
48642	2561	28	16	390	25	458	
48643	2557	23	16	390	25	433	
48644	2554	10	16	391	25	408	
48645	2554	11	17	395	26	384	
48646	2551	9	16	400	25	361	

5 rows × 55 columns

```
data.fillna(0)
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hill
0	2596	51	3	258	0	510	
1	2590	56	2	212	-6	390	
2	2804	139	9	268	65	3180	
3	2785	155	18	242	118	3090	
4	2595	45	2	153	-1	391	
...	...	...	...	...	...	...	
48642	2561	28	16	390	25	458	
48643	2557	23	16	390	25	433	
48644	2554	10	16	391	25	408	
48645	2554	11	17	395	26	384	
48646	2551	9	16	400	25	361	

48647 rows × 55 columns

```
data.shape
```

```
(48647, 55)
```

```
data.info()
```

```
---  ------                              --------------  -----
 0   Elevation                           48647 non-null  int64
 1   Aspect                              48647 non-null  int64
 2   Slope                               48647 non-null  int64
 3   Horizontal_Distance_To_Hydrology    48647 non-null  int64
 4   Vertical_Distance_To_Hydrology      48647 non-null  int64
 5   Horizontal_Distance_To_Roadways     48647 non-null  int64
 6   Hillshade_9am                       48647 non-null  int64
 7   Hillshade_Noon                      48647 non-null  int64
 8   Hillshade_3pm                       48647 non-null  int64
 9   Horizontal_Distance_To_Fire_Points  48647 non-null  int64
 10  Wilderness_Area1                    48647 non-null  int64
 11  Wilderness_Area2                    48647 non-null  int64
 12  Wilderness_Area3                    48647 non-null  int64
 13  Wilderness_Area4                    48646 non-null  float64
 14  Soil_Type1                          48646 non-null  float64
 15  Soil_Type2                          48646 non-null  float64
 16  Soil_Type3                          48646 non-null  float64
 17  Soil_Type4                          48646 non-null  float64
 18  Soil_Type5                          48646 non-null  float64
 19  Soil_Type6                          48646 non-null  float64
 20  Soil_Type7                          48646 non-null  float64
 21  Soil_Type8                          48646 non-null  float64
 22  Soil_Type9                          48646 non-null  float64
 23  Soil_Type10                         48646 non-null  float64
 24  Soil_Type11                         48646 non-null  float64
 25  Soil_Type12                         48646 non-null  float64
 26  Soil_Type13                         48646 non-null  float64
 27  Soil_Type14                         48646 non-null  float64
 28  Soil_Type15                         48646 non-null  float64
 29  Soil_Type16                         48646 non-null  float64
 30  Soil_Type17                         48646 non-null  float64
 31  Soil_Type18                         48646 non-null  float64
 32  Soil_Type19                         48646 non-null  float64
 33  Soil_Type20                         48646 non-null  float64
 34  Soil_Type21                         48646 non-null  float64
 35  Soil_Type22                         48646 non-null  float64
 36  Soil_Type23                         48646 non-null  float64
 37  Soil_Type24                         48646 non-null  float64
 38  Soil_Type25                         48646 non-null  float64
 39  Soil_Type26                         48646 non-null  float64
 40  Soil_Type27                         48646 non-null  float64
 41  Soil_Type28                         48646 non-null  float64
 42  Soil_Type29                         48646 non-null  float64
 43  Soil_Type30                         48646 non-null  float64
 44  Soil_Type31                         48646 non-null  float64
```

```
 49  Soil_Type36                        48646 non-null  float64
 50  Soil_Type37                        48646 non-null  float64
 51  Soil_Type38                        48646 non-null  float64
 52  Soil_Type39                        48646 non-null  float64
 53  Soil_Type40                        48646 non-null  float64
 54  Cover_Type                         48646 non-null  float64
dtypes: float64(42), int64(13)
memory usage: 20.4 MB
```

```
data.describe()
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To
count	48647.000000	48647.000000	48647.000000	48647.000000	48647.000000	486
mean	2831.526672	141.228544	13.172508	239.109606	39.873538	32
std	284.219268	106.378387	7.501714	204.272413	50.321057	19
min	1863.000000	0.000000	0.000000	0.000000	-146.000000	
25%	2674.000000	56.000000	8.000000	85.000000	6.000000	13
50%	2854.000000	112.000000	12.000000	190.000000	24.000000	30
75%	3021.000000	207.000000	17.000000	342.000000	58.000000	50
max	3849.000000	360.000000	61.000000	1343.000000	554.000000	7

8 rows × 55 columns

```
data.columns
```

```
Index(['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology',
       'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways',
       'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
       'Horizontal_Distance_To_Fire_Points', 'Wilderness_Area1',
       'Wilderness_Area2', 'Wilderness_Area3', 'Wilderness_Area4',
       'Soil_Type1', 'Soil_Type2', 'Soil_Type3', 'Soil_Type4', 'Soil_Type5',
       'Soil_Type6', 'Soil_Type7', 'Soil_Type8', 'Soil_Type9', 'Soil_Type10',
       'Soil_Type11', 'Soil_Type12', 'Soil_Type13', 'Soil_Type14',
       'Soil_Type15', 'Soil_Type16', 'Soil_Type17', 'Soil_Type18',
       'Soil_Type19', 'Soil_Type20', 'Soil_Type21', 'Soil_Type22',
       'Soil_Type23', 'Soil_Type24', 'Soil_Type25', 'Soil_Type26',
       'Soil_Type27', 'Soil_Type28', 'Soil_Type29', 'Soil_Type30',
       'Soil_Type31', 'Soil_Type32', 'Soil_Type33', 'Soil_Type34',
       'Soil_Type35', 'Soil_Type36', 'Soil_Type37', 'Soil_Type38',
       'Soil_Type39', 'Soil_Type40', 'Cover_Type'],
      dtype='object')
```

```
X = data.drop(columns='Cover_Type')
y = data['Cover_Type']
```

```
data = data.dropna(subset=['Cover_Type'])
X = data.drop(columns='Cover_Type')
y = data['Cover_Type']
```

```
columns_to_replace_zeros = ['Elevation',    'Aspect' ,  'Slope' ,   'Horizontal_Distance_To_Hydrology', 'Vertical_Distance_To_Hydrology',

for column in columns_to_replace_zeros:
    median_val = X[column].dropna().median()
    X[column] = X[column].replace(0, median_val)
for column in X.columns:
    X[column] = X[column].fillna(X[column].median())

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

#linear regression
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-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-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression
LogisticRegression(random_state=42)
```

```
X[columns_to_replace_zeros] = X[columns_to_replace_zeros].replace(0, X[columns_to_replace_zeros].median())
```

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

```
y_pred_lr = lr_model.predict(X_test) # This line is missing, causing the error

lr_report = classification_report(y_test, y_pred_lr,output_dict=True)
lr_accuracy = accuracy_score(y_test, y_pred_lr)
print("Logistic Regression Report:\n", lr_report)
print("Accuracy:", lr_accuracy)
```

```
Logistic Regression Report:
 {'1.0': {'precision': 0.7701582516955539, 'recall': 0.5331246739697444, 'f1-score': 0.6300863131935882, 'support': 1917.0}, '2.0': {'pr
Accuracy: 0.8060637204522096
```

```
#naive bias classification
```

```
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
y_pred_nb = nb_model.predict(X_test)
```

```
if isinstance(y_test, pd.Series):
    y_test = y_test.fillna(y_test.mode()[0])
elif isinstance(y_test, np.ndarray):
    most_frequent_value = np.nanmax(y_test)
    y_test = np.nan_to_num(y_test, nan=most_frequent_value)
nb_report = classification_report(y_test, y_pred_nb, output_dict=True)
nb_accuracy = accuracy_score(y_test, y_pred_nb)
print("Naive Bayes Report:\n", nb_report)
print("Accuracy:", nb_accuracy)
```

```
Naive Bayes Report:
 {'1.0': {'precision': 0.16946080652469417, 'recall': 0.1950965049556599, 'f1-score': 0.1813773035887488, 'support': 1917.0}, '2.0': {'p
Accuracy: 0.18191161356628982
```

```
#KNN CLASSSIFICATION
```

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred_knn = knn_model.predict(X_test)
```

```
knn_report = classification_report(y_test, y_pred_knn,output_dict=True)
knn_accuracy = accuracy_score(y_test, y_pred_knn)
```

```
print("KNN Report:\n", knn_report)
print("Accuracy:", knn_accuracy)
```

```
KNN Report:
 {'1.0': {'precision': 0.8480392156862745, 'recall': 0.812206572769953, 'f1-score': 0.829736211031175, 'support': 1917.0}, '2.0': {'prec
Accuracy: 0.8858170606372046
```

```
#DECISION TREE CLASSSIFICATION
```

```
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
y_pred_dt = dt_model.predict(X_test)
```

```
dt_report = classification_report(y_test, y_pred_dt,output_dict=True)
dt_accuracy = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Report:\n", dt_report)
print("Accuracy:", dt_accuracy)
```

```
Decision Tree Report:
 {'1.0': {'precision': 0.8709508881922675, 'recall': 0.8695878977569118, 'f1-score': 0.8702688593056643, 'support': 1917.0}, '2.0': {'pr
Accuracy: 0.9087358684480986
```

```
#RANDOM FOREST
```

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
y_pred_rf = rf_model.predict(X_test)
```

```
rf_report = classification_report(y_test, y_pred_rf,output_dict=True)
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print("Random Forest Report:\n", rf_report)
print("Accuracy:", rf_accuracy)
```

```
Random Forest Report:
 {'1.0': {'precision': 0.9238410596026491, 'recall': 0.8732394366197183, 'f1-score': 0.8978278358809332, 'support': 1917.0}, '2.0': {'pr
Accuracy: 0.9328879753340185
```

```
# k-means cluster for classification
```

```
from sklearn.metrics import adjusted_rand_score
```

```
kmeans_model = KMeans(n_clusters=2, random_state=42)
kmeans_model.fit(X_train)
```

```
KMeans
KMeans(n_clusters=2, random_state=42)
```

```
y_pred_kmeans = kmeans_model.predict(X_test)
```

```
ari_score = adjusted_rand_score(y_test, y_pred_kmeans)
print("K-Means ARI Score:", ari_score)
```

```
K-Means ARI Score: 0.02416873487204871
```

```
# comparision table ( basically comparing all the)


results = {
      "Model": ["Logistic Regression", "Naive Bayes", "KNN", "Decision Tree", "Random Forest"],
    "Accuracy": [lr_accuracy, nb_accuracy, knn_accuracy, dt_accuracy, rf_accuracy],  # Ensure these variables are defined
}

comparison_df = pd.DataFrame(results)
print(comparison_df)
```

```
                 Model  Accuracy
0  Logistic Regression  0.806064
1          Naive Bayes  0.181912
2                  KNN  0.885817
3        Decision Tree  0.908736
4        Random Forest  0.932888
```

```
# this is the normal overall comparision


# detaild comparision table


models_performance = {
    "Model": ["Logistic Regression", "Naive Bayes", "KNN", "Decision Tree", "Random Forest"],
    "Precision (Class 1)": [
        lr_report['1.0']['precision'], nb_report['1.0']['precision'], # Changed '1' to '1.0' to match the actual key in the dictionaries
        knn_report['1.0']['precision'], dt_report['1.0']['precision'],
        rf_report['1.0']['precision']
    ],
    "Recall (Class 1)": [
        lr_report['1.0']['recall'], nb_report['1.0']['recall'], # Changed '1' to '1.0'
        knn_report['1.0']['recall'], dt_report['1.0']['recall'],
        rf_report['1.0']['recall']
    ],
    "F1-Score (Class 1)": [
        lr_report['1.0']['f1-score'], nb_report['1.0']['f1-score'], # Changed '1' to '1.0'
        knn_report['1.0']['f1-score'], dt_report['1.0']['f1-score'],
        rf_report['1.0']['f1-score']
    ],
    "Accuracy": [
        lr_accuracy, nb_accuracy, knn_accuracy,
        dt_accuracy, rf_accuracy
    ]
}

comparison_df = pd.DataFrame(models_performance)
print(comparison_df)
```

```
                 Model  Precision (Class 1)  Recall (Class 1)  \
0  Logistic Regression             0.770158          0.533125
1          Naive Bayes             0.169461          0.195097
2                  KNN             0.848039          0.812207
3        Decision Tree             0.870951          0.869588
4        Random Forest             0.923841          0.873239

   F1-Score (Class 1)  Accuracy
0            0.630086  0.806064
1            0.181377  0.181912
2            0.829736  0.885817
3            0.870269  0.908736
4            0.897828  0.932888
```

```
#RESULT:- THE  BEST IS RANDOM FOREST AFTER COMPARING
```