In [177…
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [180…
```
adult=pd.read_csv(r"D:\adult.data.csv",na_values='?',skipinitialspace=True)
adult
```

Out[180…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	na co
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	Un S
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	Un S
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	Un S
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	Un S
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
48837	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female	0	0	36	Un S
48838	64	NaN	321403	HS-grad	9	Widowed	NaN	Other-relative	Black	Male	0	0	40	Un S
48839	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	Un S
48840	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455	0	40	Un S
48841	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	Un S

48842 rows × 15 columns

In [182…
```
adult.head()
```

Out[182…

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba

In [184…
```
adult.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             48842 non-null  int64
 1   workclass       46043 non-null  object
 2   fnlwgt          48842 non-null  int64
 3   education       48842 non-null  object
 4   education-num   48842 non-null  int64
 5   marital-status  48842 non-null  object
 6   occupation      46033 non-null  object
 7   relationship    48842 non-null  object
 8   race            48842 non-null  object
 9   sex             48842 non-null  object
 10  capital-gain    48842 non-null  int64
 11  capital-loss    48842 non-null  int64
 12  hours-per-week  48842 non-null  int64
 13  native-country  47985 non-null  object
 14  income          48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

In [186…
```
adult.shape
```

Out[186…
```
(48842, 15)
```

In [188…
```
adult.dtypes
```

Out[188…
```
age               int64
workclass        object
fnlwgt            int64
education        object
education-num     int64
marital-status   object
occupation       object
relationship     object
race             object
sex              object
capital-gain      int64
capital-loss      int64
hours-per-week    int64
native-country   object
income           object
dtype: object
```

In [190…
```
adult.isna().sum()
```

Out[190…
```
age                0
workclass       2799
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation      2809
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country   857
income             0
dtype: int64
```

In [192…
```
adult_new=adult.fillna(method='ffill')
adult_new
```

Out[192...

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	na co
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	Un S
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	Un S
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	Un S
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	Un S
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
48837	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female	0	0	36	Un S
48838	64	Private	321403	HS-grad	9	Widowed	Prof-specialty	Other-relative	Black	Male	0	0	40	Un S
48839	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	Un S
48840	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455	0	40	Un S
48841	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	Un S

48842 rows × 15 columns

In [194...
```
adult_new.columns
```

Out[194...
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'income'],
      dtype='object')
```

In [196...
```
adult_new=adult_new.drop(columns=['fnlwgt','relationship','native-country'],axis=1)
adult_new
```

Out[196...

	age	workclass	education	education-num	marital-status	occupation	race	sex	capital-gain	capital-loss	hours-per-week	income
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	White	Male	2174	0	40	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	White	Male	0	0	13	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	White	Male	0	0	40	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Black	Male	0	0	40	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Black	Female	0	0	40	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...
48837	39	Private	Bachelors	13	Divorced	Prof-specialty	White	Female	0	0	36	<=50K.
48838	64	Private	HS-grad	9	Widowed	Prof-specialty	Black	Male	0	0	40	<=50K.
48839	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	White	Male	0	0	50	<=50K.
48840	44	Private	Bachelors	13	Divorced	Adm-clerical	Asian-Pac-Islander	Male	5455	0	40	<=50K.
48841	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	White	Male	0	0	60	>50K.

48842 rows × 12 columns

In [198...
```
adult_new.describe()
```

Out[198...

	age	education-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	10.078089	1079.067626	87.502314	40.422382
std	13.710510	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	48.000000	12.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	4356.000000	99.000000

In [200...
```
adult_new.isna().sum()
```

Out[200...
```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
income            0
dtype: int64
```

In [202...
```
adult_new['income']=adult_new['income'].str.strip().str.replace('.','',regex=False)
adult_new
```

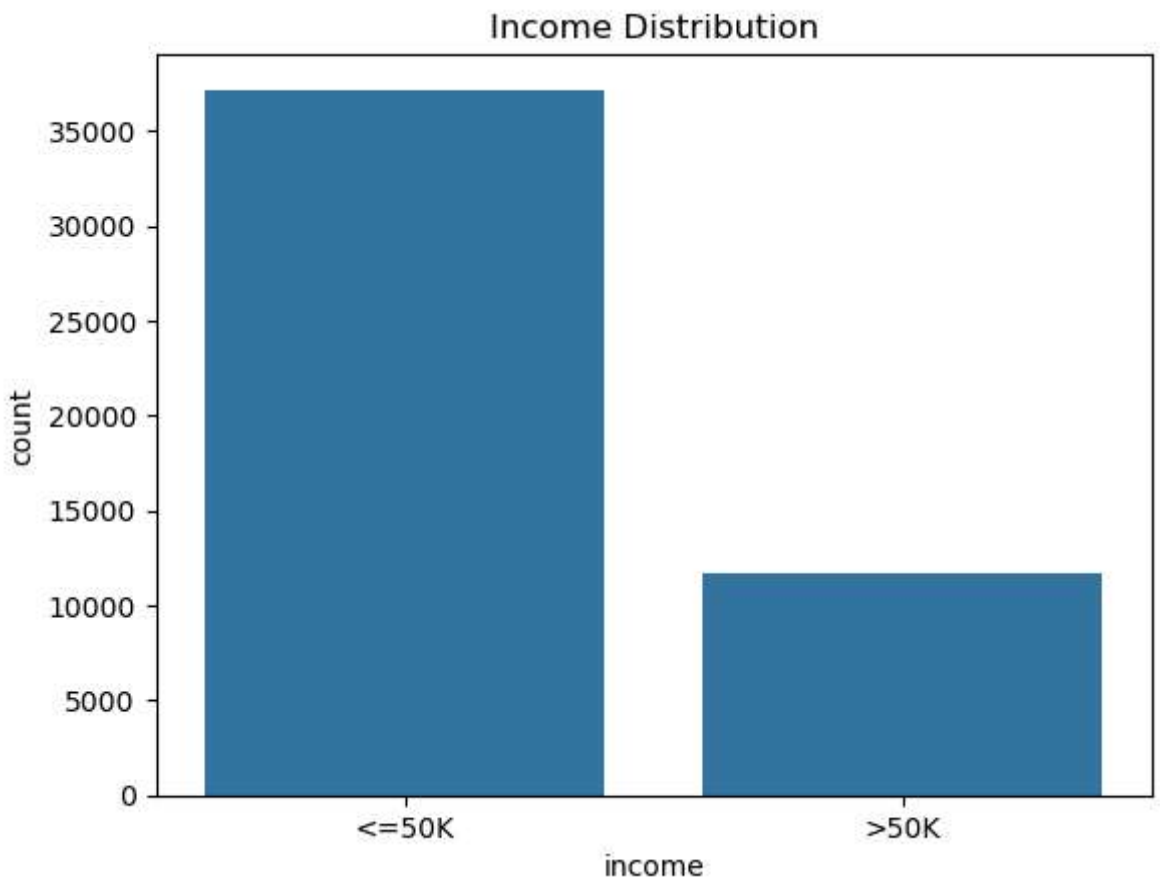
Out[202...

	age	workclass	education	education-num	marital-status	occupation	race	sex	capital-gain	capital-loss	hours-per-week	income
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	White	Male	2174	0	40	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	White	Male	0	0	13	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	White	Male	0	0	40	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Black	Male	0	0	40	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Black	Female	0	0	40	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...
48837	39	Private	Bachelors	13	Divorced	Prof-specialty	White	Female	0	0	36	<=50K
48838	64	Private	HS-grad	9	Widowed	Prof-specialty	Black	Male	0	0	40	<=50K
48839	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	White	Male	0	0	50	<=50K
48840	44	Private	Bachelors	13	Divorced	Adm-clerical	Asian-Pac-Islander	Male	5455	0	40	<=50K
48841	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	White	Male	0	0	60	>50K

48842 rows × 12 columns

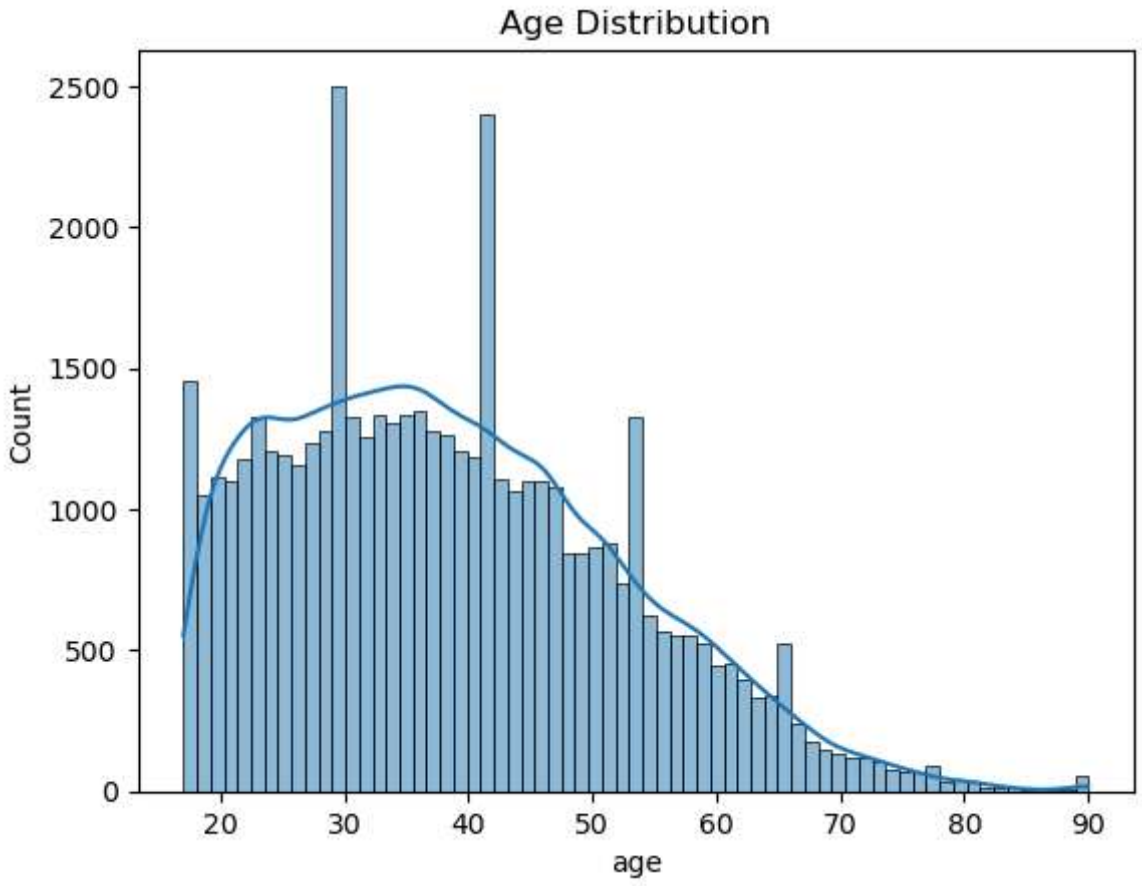
In [204...
```
sns.countplot(x='income',data=adult_new)
plt.title('Income Distribution')
```

Out[204... `Text(0.5, 1.0, 'Income Distribution')`



In [205...
```
sns.histplot(adult_new['age'],kde=True)
plt.title('Age Distribution')
```

Out[205... `Text(0.5, 1.0, 'Age Distribution')`



In [210…
```
from sklearn.preprocessing import LabelEncoder
```

In [212…
```
encode=LabelEncoder()
adult_new['income']=encode.fit_transform(adult_new['income'])
adult_new['income']
```

Out[212…
```
0        0
1        0
2        0
3        0
4        0
        ..
48837    0
48838    0
48839    0
48840    0
48841    1
Name: income, Length: 48842, dtype: int32
```

In [214…
```
adult_new.columns
```

Out[214…
```
Index(['age', 'workclass', 'education', 'education-num', 'marital-status',
       'occupation', 'race', 'sex', 'capital-gain', 'capital-loss',
       'hours-per-week', 'income'],
      dtype='object')
```

In [216…
```
adult_new=pd.get_dummies(adult_new,drop_first=True)
adult_new
```

Out[216…

	age	education-num	capital-gain	capital-loss	hours-per-week	income	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Self-emp-inc	…	occu
0	39	13	2174	0	40	0	False	False	False	False	…	
1	50	13	0	0	13	0	False	False	False	False	…	
2	38	9	0	0	40	0	False	False	True	False	…	
3	53	7	0	0	40	0	False	False	True	False	…	
4	28	13	0	0	40	0	False	False	True	False	…	
…	…	…	…	…	…	…	…	…	…	…	…	…
48837	39	13	0	0	36	0	False	False	True	False	…	
48838	64	9	0	0	40	0	False	False	True	False	…	
48839	38	13	0	0	50	0	False	False	True	False	…	
48840	44	13	5455	0	40	0	False	False	True	False	…	
48841	35	13	0	0	60	1	False	False	False	True	…	

48842 rows × 52 columns

In [228…
```
x=adult_new.drop(columns=['income'])
x
```

Out[228...

	age	education-num	capital-gain	capital-loss	hours-per-week	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Self-emp-inc	workclass_Self-emp-not-inc
0	39	13	2174	0	40	False	False	False	False	False
1	50	13	0	0	13	False	False	False	False	True
2	38	9	0	0	40	False	False	True	False	False
3	53	7	0	0	40	False	False	True	False	False
4	28	13	0	0	40	False	False	True	False	False
...	...	...	...	...	...	...	...	...	...	...
48837	39	13	0	0	36	False	False	True	False	False
48838	64	9	0	0	40	False	False	True	False	False
48839	38	13	0	0	50	False	False	True	False	False
48840	44	13	5455	0	40	False	False	True	False	False
48841	35	13	0	0	60	False	False	False	True	False

48842 rows × 51 columns

In [230...
```
y=adult_new['income']
y
```

Out[230...
```
0        0
1        0
2        0
3        0
4        0
        ..
48837    0
48838    0
48839    0
48840    0
48841    1
Name: income, Length: 48842, dtype: int32
```

In [258...
```
from sklearn.model_selection import train_test_split
```

In [260...
```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=42,stratify=y)
```

In [262...
```
x_test.head()
```

Out[262...

	age	education-num	capital-gain	capital-loss	hours-per-week	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Self-emp-inc	workclass_Self-emp-not-inc
40421	30	9	0	0	40	False	False	True	False	False
47738	54	12	0	0	39	False	False	False	False	False
518	21	10	0	0	35	False	False	True	False	False
8564	35	9	2885	0	40	False	False	True	False	False
31355	42	10	0	0	45	True	False	False	False	False

5 rows × 51 columns

In [264...
```
x_train.shape
```

Out[264...
```
(39073, 51)
```

In [266...
```
x_test.shape
```

Out[266...
```
(9769, 51)
```

In [269...
```
from sklearn.preprocessing import StandardScaler
```

In [271...
```
scaler=StandardScaler()
scaler
```

Out[271...
```
StandardScaler()
```

In [273…
```
scaled_x_train=scaler.fit_transform(x_train)
scaled_x_train
```

Out[273…
```
array([[-0.12009008,  1.13821044, -0.14407503, ..., -0.09327335,
         0.4115339 , -1.42415255],
       [ 1.26824482, -0.80402454, -0.14407503, ..., -0.09327335,
         0.4115339 ,  0.70217197],
       [ 1.04903405, -0.41557754, -0.14407503, ..., -0.09327335,
         0.4115339 ,  0.70217197],
       ...,
       [ 1.63359611, -0.02713055, -0.14407503, ..., -0.09327335,
         0.4115339 ,  0.70217197],
       [-0.04701982,  0.36131645, -0.14407503, ..., -0.09327335,
         0.4115339 , -1.42415255],
       [ 0.31833147, -0.02713055, -0.14407503, ..., -0.09327335,
        -2.42993349,  0.70217197]])
```

In [275…
```
scaled_x_test=scaler.fit_transform(x_test)
scaled_x_test
```

Out[275…
```
array([[-0.62594162, -0.43451805, -0.14860513, ..., -0.08434099,
         0.41257776,  0.71243367],
       [ 1.11198537,  0.73872873, -0.14860513, ..., -0.08434099,
         0.41257776,  0.71243367],
       [-1.27766425, -0.04343579, -0.14860513, ..., -0.08434099,
         0.41257776, -1.40363945],
       ...,
       [ 1.4740535 , -0.43451805, -0.14860513, ..., -0.08434099,
         0.41257776,  0.71243367],
       [-0.69835525, -0.04343579, -0.14860513, ..., -0.08434099,
         0.41257776, -1.40363945],
       [ 1.4740535 , -0.04343579, -0.14860513, ..., -0.08434099,
        -2.42378549,  0.71243367]])
```

Logistic Regression

In [279…
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
```

In [281…
```
model=LogisticRegression(solver='liblinear')
model
```

Out[281…
```
LogisticRegression(solver='liblinear')
```

In [283…
```
model.fit(x_train,y_train)
```

Out[283…
```
LogisticRegression(solver='liblinear')
```

In [285…
```
y_pred=model.predict(x_test)
y_pred
```

Out[285…
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [289…
```
accuracy_log=accuracy_score(y_test,y_pred)
accuracy_log
```

Out[289…
```
0.8497287337496161
```

In [291…
```
print("Accuracy :",accuracy_log)
```

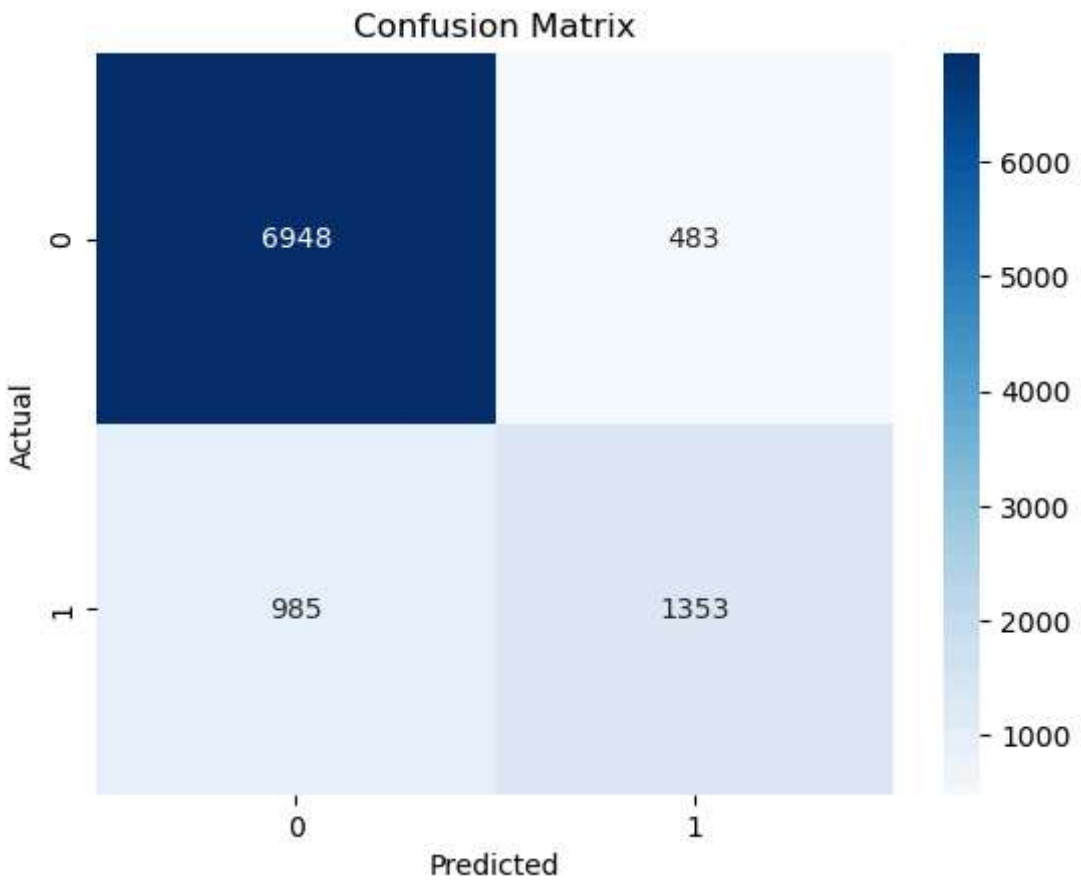
```
Accuracy : 0.8497287337496161
```

In [294…
```
confusion_matrix(y_test,y_pred)
```

Out[294…
```
array([[6948,  483],
       [ 985, 1353]], dtype=int64)
```

In [296…
```
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d',cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

Out[296…
```
Text(50.722222222222214, 0.5, 'Actual')
```



In [297…
```
print("classification Report :\n")
print(classification_report(y_test,y_pred))
```

```
classification Report :

              precision    recall  f1-score   support

           0       0.88      0.94      0.90      7431
           1       0.74      0.58      0.65      2338

    accuracy                           0.85      9769
   macro avg       0.81      0.76      0.78      9769
weighted avg       0.84      0.85      0.84      9769
```

Random Forest

In [301…
```
from sklearn.ensemble import RandomForestClassifier
```

In [303…
```
model=RandomForestClassifier(n_estimators=50,random_state=100)
model
```

Out[303…
```
RandomForestClassifier(n_estimators=50, random_state=100)
```

In [305…
```
model.fit(x_train,y_train)
```

Out[305…
```
RandomForestClassifier(n_estimators=50, random_state=100)
```

In [306…
```
y_pred=model.predict(x_test)
y_pred
```

Out[306…
```
array([0, 0, 0, ..., 1, 0, 0])
```

In [310…
```
accuracy_rand=accuracy_score(y_test,y_pred)
accuracy_rand
print("Accuracy :",accuracy_rand)
```

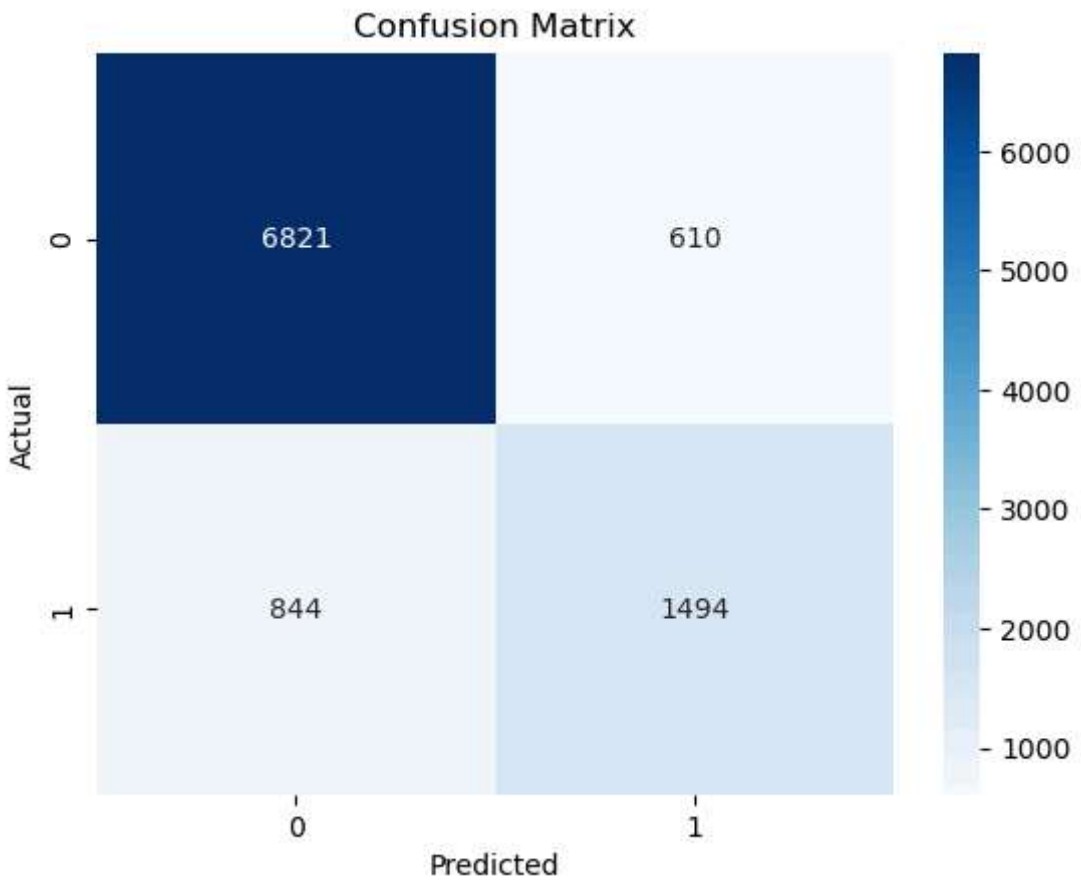
```
Accuracy : 0.8511618384686253
```

In [313…
```
confusion_matrix(y_test,y_pred)
```

Out[313…
```
array([[6821,  610],
       [ 844, 1494]], dtype=int64)
```

In [315…
```
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d',cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

Out[315…
```
Text(50.722222222222214, 0.5, 'Actual')
```



In [317…
```
print("classification Report :\n")
print(classification_report(y_test,y_pred))
```

```
classification Report :

              precision    recall  f1-score   support

           0       0.89      0.92      0.90      7431
           1       0.71      0.64      0.67      2338

    accuracy                           0.85      9769
   macro avg       0.80      0.78      0.79      9769
weighted avg       0.85      0.85      0.85      9769
```

Decision tree classification

In [320…
```
from sklearn.tree import DecisionTreeClassifier
```

In [322…
```
model=DecisionTreeClassifier(criterion='gini',max_depth=3)
model
```

Out[322…
```
DecisionTreeClassifier(max_depth=3)
```

In [324…
```
model.fit(x_train,y_train)
```

Out[324…
```
DecisionTreeClassifier(max_depth=3)
```

In [326…
```
y_pred=model.predict(x_test)
y_pred
```

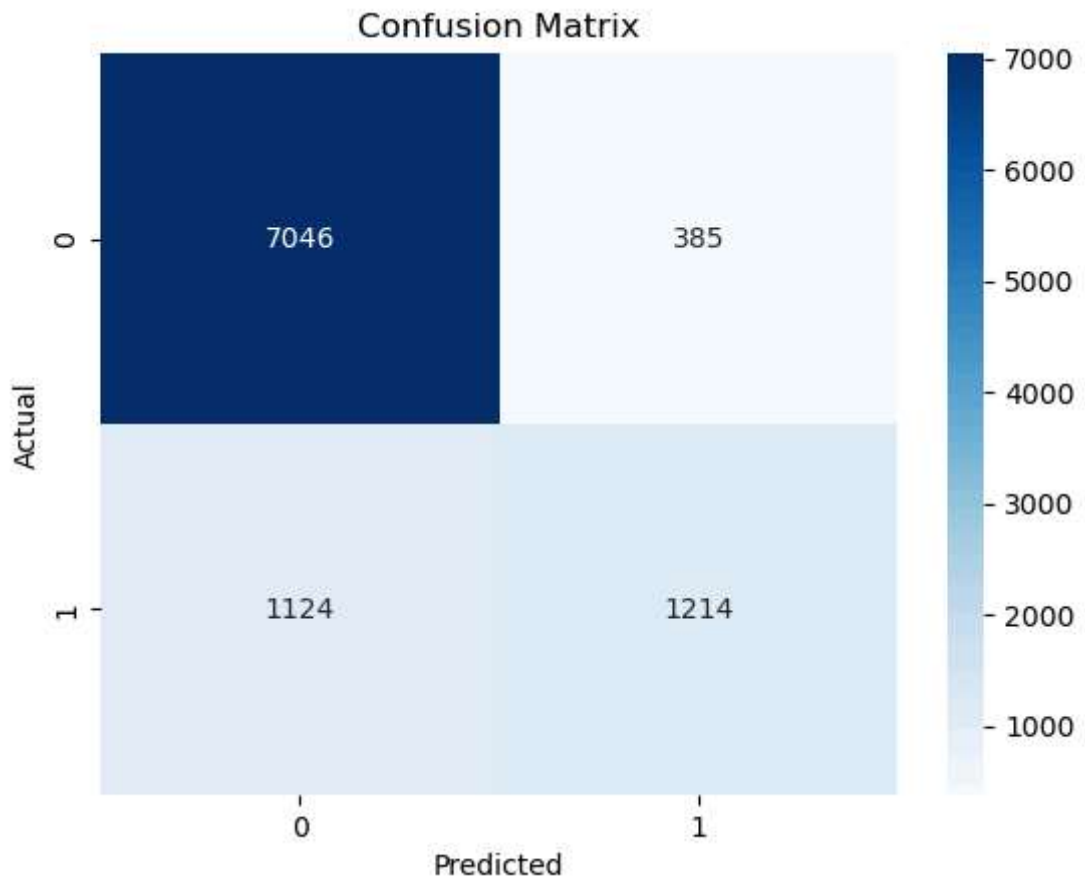
Out[326…
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [329…
```
confusion_matrix(y_test,y_pred)
```

Out[329…
```
array([[7046,  385],
       [1124, 1214]], dtype=int64)
```

In [331…
```
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d',cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

Out[331…
```
Text(50.722222222222214, 0.5, 'Actual')
```



In [334…
```
accuracy_tree=accuracy_score(y_test,y_pred)
accuracy_tree
```

Out[334…
```
0.8455317842153751
```

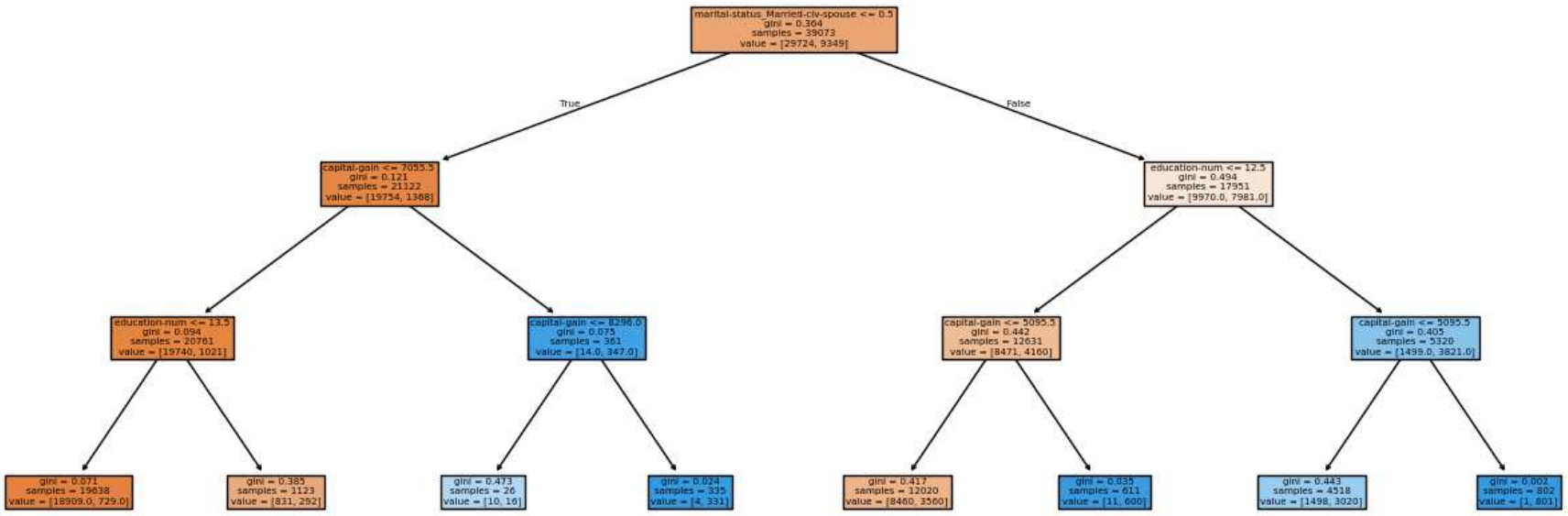
In [336…
```
print("classification Report :\n")
print(classification_report(y_test,y_pred))
```

```
classification Report :

              precision    recall  f1-score   support

           0       0.86      0.95      0.90      7431
           1       0.76      0.52      0.62      2338

    accuracy                           0.85      9769
   macro avg       0.81      0.73      0.76      9769
weighted avg       0.84      0.85      0.83      9769
```

Tree ploting

In [339…
```
from sklearn.tree import plot_tree
```

In [341…
```
plt.figure(figsize=(16,6))
plot_tree(model,feature_names=x.columns,filled=True)
plt.show()
```



K-Nearest Neighbors (KNN)

In [344…
```
from sklearn.neighbors import KNeighborsClassifier
```

In [346…
```
model=KNeighborsClassifier(n_neighbors=3)
model
```

Out[346…

KNeighborsClassifier

KNeighborsClassifier(n_neighbors=3)

In [348…

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

Out[348…

KNeighborsClassifier

KNeighborsClassifier(n_neighbors=3)

In [350…

```
y_pred=model.predict(scaled_x_test)
y_pred
```

Out[350…

```
array([0, 0, 0, ..., 0, 0, 0])
```

Confusion matrix

In [352…

```
confusion_matrix(y_test,y_pred)
```

Out[352…

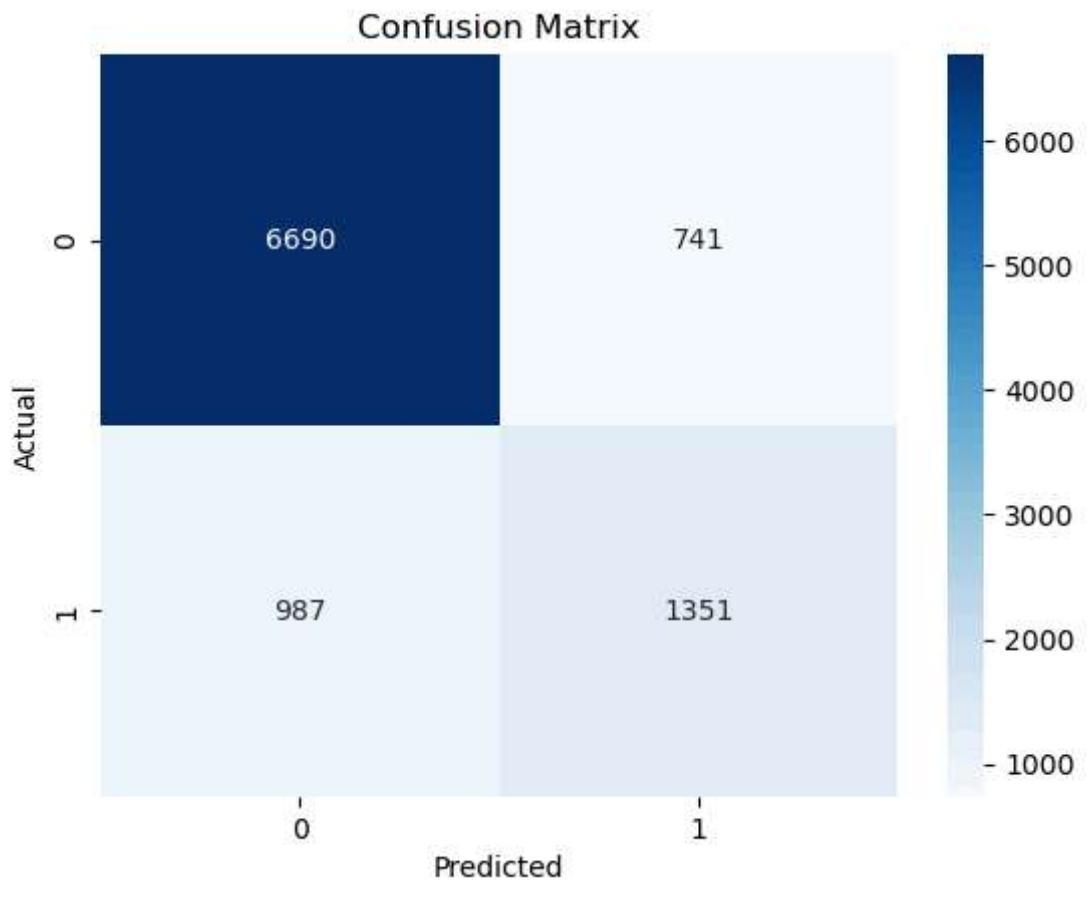
```
array([[6690,  741],
       [ 987, 1351]], dtype=int64)
```

In [353…

```
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d',cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
```

Out[353…

```
Text(50.722222222222214, 0.5, 'Actual')
```



In [355…

```
accuracy_knn=accuracy_score(y_test,y_pred)
accuracy_knn
```

Out[355…

```
0.8231139318251612
```

In [356…

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.90      0.89      7431
           1       0.65      0.58      0.61      2338

    accuracy                           0.82      9769
   macro avg       0.76      0.74      0.75      9769
weighted avg       0.82      0.82      0.82      9769
```

Collect accuracy scores from all models

In [361…

```
Model_name={"LogisticRegression":round(accuracy_log,2),
        "RandomForestClassifier":round(accuracy_rand,2),
        "DecisionTreeClassifier":round(accuracy_tree,2),
        "KNearestNeighbors":round(accuracy_knn,2) }
Model_name
```

Out[361…
```
{'LogisticRegression': 0.85,
 'RandomForestClassifier': 0.85,
 'DecisionTreeClassifier': 0.85,
 'KNearestNeighbors': 0.82}
```

Create a Dataframe for vizualization

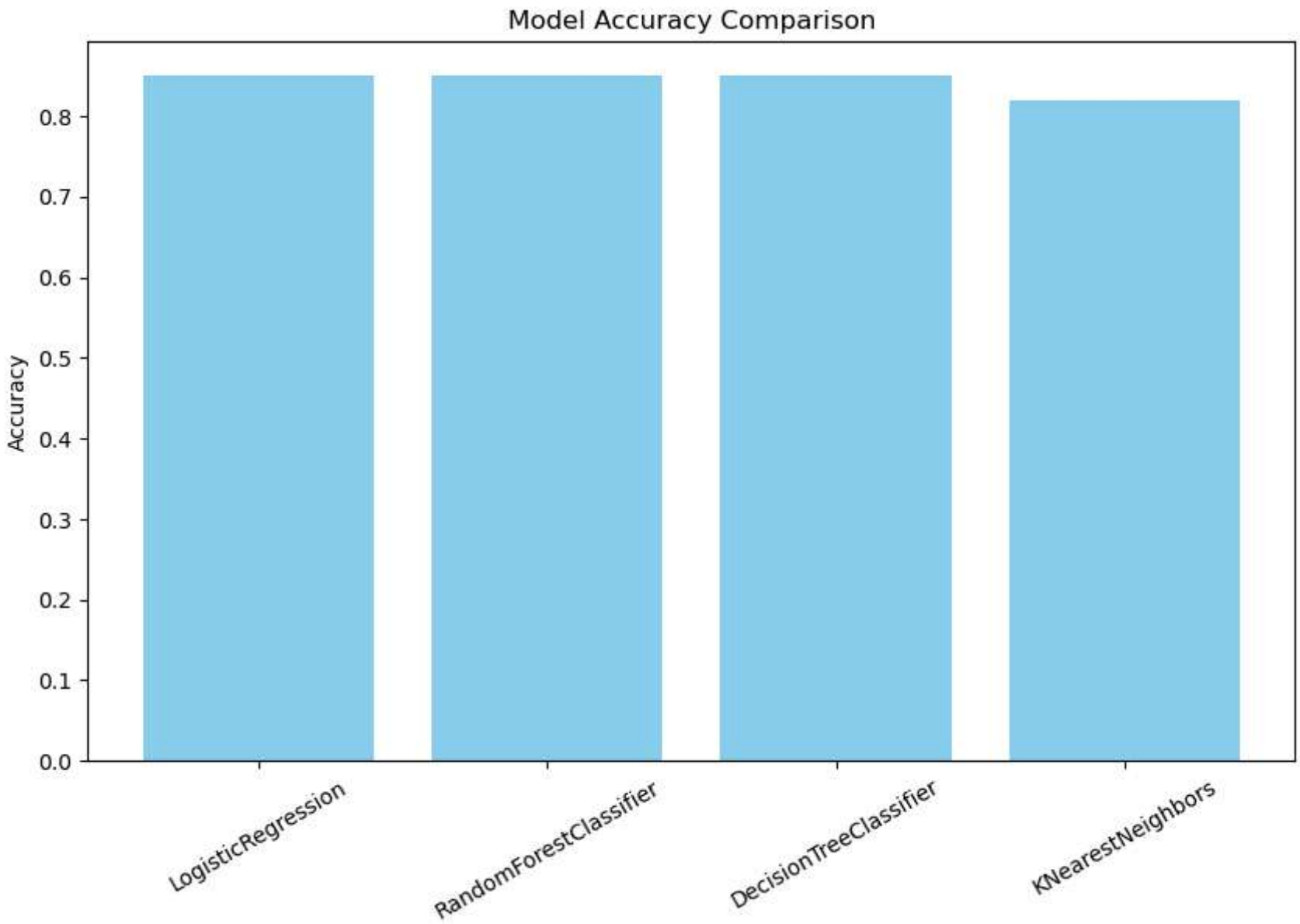
In [366…
```
project_report=pd.DataFrame(list(Model_name.items()),columns=["Model","Accuracy"])
project_report
```

Out[366…

	Model	Accuracy
0	LogisticRegression	0.85
1	RandomForestClassifier	0.85
2	DecisionTreeClassifier	0.85
3	KNearestNeighbors	0.82

In [368…
```
import matplotlib.pyplot as plt
```

In [370…
```
Model_name=['LogisticRegression','RandomForestClassifier','DecisionTreeClassifier','KNearestNeighbors']
Accuracies=[round(accuracy_log,2),round(accuracy_rand,2),round(accuracy_tree,2),round(accuracy_knn,2)]
plt.figure(figsize=(10,6))
plt.title('Model Accuracy Comparison')
plt.bar(Model_name,Accuracies,color='skyblue')
plt.ylabel('Accuracy')
plt.xticks(rotation=30)
plt.show()
```



The Logistic Regression, Random forest classifier and Decision tree classifier gave about 85% accuracy,while KNN gave 82% accuracy.

The Random forest as the best performing model. Because it had the highest F1 score, which balances both precision and recall.