Practical 5 : Use the dataset named "People Charm case.csv" that deals with HR analytics and answer the following questions:

1. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as The accuracy score for the predicted model is?

2. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as 2 and find out how many samples are misclassified?

3. Build a k-Nearest Neighbors model using all the variables. Use 75% of the data as the training set, fix the random state as 0 and the k value as 2.The accuracy score for the predicted model is?

##Use the dataset named "People Charm case.csv" that deals with HR analytics and answer the following questions

##1. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as The accuracy score for the predicted model is?

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv('People Charm case.csv')

# Define features and target
X = df.drop(columns='numberOfProjects')  # Replace 'target' with your actual target column name
y = df['numberOfProjects']  # Replace 'target' with your actual target column name

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Create a preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X.select_dtypes(include=['int64', 'float64']).columns),
        ('cat', OneHotEncoder(), X.select_dtypes(include=['object']).columns)
    ])

# Create a logistic regression model
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42))
])

# Train the model
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy score: {accuracy:.4f}')
```

2. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as 2 and find out how many samples are misclassified?

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import confusion_matrix
import numpy as np

# Define your features (X) and target (y)
X = data.drop(columns=['numberOfProjects'])  # Replace 'target_column' with the actual target column name
y = data['numberOfProjects']  # Replace 'target_column' with the actual target column name

# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

# Define the preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns)
    ],
    remainder='passthrough'  # Keep other columns as they are
)
```

```
# Define the pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000, random_state=2))
])

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=2)

# Train the model
pipeline.fit(X_train, y_train)

# Make predictions
y_pred = pipeline.predict(X_test)

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Calculate the number of misclassified samples
misclassified_samples = np.sum(cm) - np.trace(cm)
print(f"Number of misclassified samples: {misclassified_samples}")
```

3. Build a k-Nearest Neighbors model using all the variables. Use 75% of the data as the training set, fix the random state as 0 and the k value as 2.The accuracy score for the predicted model is?

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

# Load your dataset (replace 'your_data.csv' with your actual file)
data = pd.read_csv('People Charm case.csv')

# Define your features (X) and target (y)
X = data.drop(columns=['numberOfProjects'])  # Replace 'target_column' with the actual target column name
y = data['numberOfProjects']  # Replace 'target_column' with the actual target column name

# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

# Define the preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns)
```

```
    ],
    remainder='passthrough'  # Keep other columns as they are
)

# Define the pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', KNeighborsClassifier(n_neighbors=2, algorithm='auto'))  # k=2
])

# Split the data into training and testing sets (75% training, 25% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

# Train the model
pipeline.fit(X_train, y_train)

# Make predictions
y_pred = pipeline.predict(X_test)

# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy score: {accuracy:.2f}")
```