1) Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

data = pd.read_csv('/content/dataset-tortuga.csv')
data.head()
```

	Unnamed: 0	NAME	USER_ID	HOURS_DATASCIENCE	HOURS_BACKEND	HOURS_FRONTEND	NUM_COURSES_BEGINNER_DATASCIENCE	NUM_COURSES_BE
0	28	Stormy Muto	58283940	7.0	39.0	29.0	2.0	
1	81	Carlos Ferro	1357218	32.0	0.0	44.0	2.0	
2	89	Robby Constantini	63212105	45.0	0.0	59.0	0.0	
3	138	Paul Mckenny	23239851	36.0	19.0	28.0	0.0	
4	143	Jean Webb	72234478	61.0	78.0	38.0	6.0	

2) Data Processing

```
# Assuming the first column 'Unnamed: 0' is an identifier and not a feature
numeric_features = data[features].select_dtypes(include=np.number).columns

imputer = SimpleImputer(strategy='mean')
data[numeric_features] = pd.DataFrame(imputer.fit_transform(data[numeric_features]), columns=numeric_features)
data.isna().sum()  # Check for missing values again
```

	0
Unnamed: 0	0
NAME	0
USER_ID	0
HOURS_DATASCIENCE	0
HOURS_BACKEND	53
HOURS_FRONTEND	16
NUM_COURSES_BEGINNER_DATASCIENCE	26
NUM_COURSES_BEGINNER_BACKEND	18
NUM_COURSES_BEGINNER_FRONTEND	39
NUM_COURSES_ADVANCED_DATASCIENCE	2
NUM_COURSES_ADVANCED_BACKEND	8
NUM_COURSES_ADVANCED_FRONTEND	37
AVG_SCORE_DATASCIENCE	220
AVG_SCORE_BACKEND	84
AVG_SCORE_FRONTEND	168
PROFILE	0

```
dtype: int64
```

3) Spliting the data

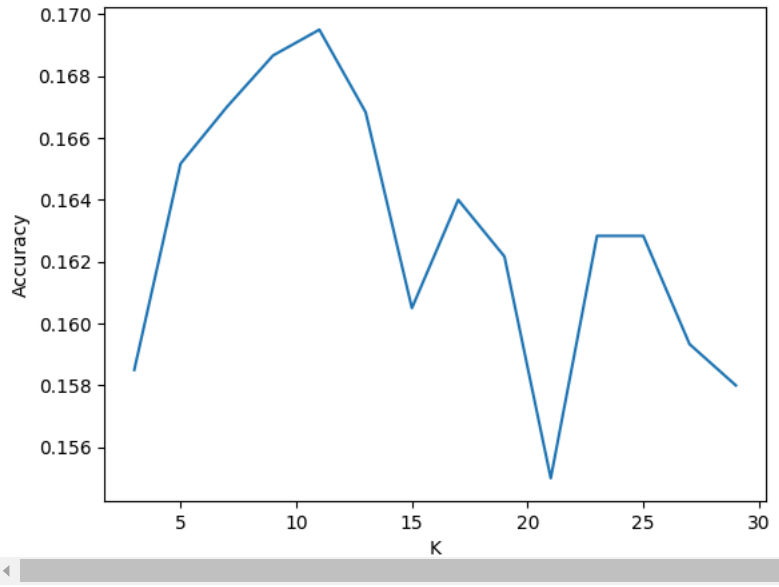
```
X = data.drop(['PROFILE'], axis = 1)
y = data['PROFILE']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

4) Scaling the data

```
# Handle non-numeric columns before applying SimpleImputer
numeric_features = X_train.select_dtypes(include=np.number).columns

# Impute missing values *before* scaling
imputer = SimpleImputer(strategy='mean')
X_train[numeric_features] = imputer.fit_transform(X_train[numeric_features])
X_test[numeric_features] = imputer.transform(X_test[numeric_features])

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train[numeric_features]) # Scale only numeric features
X_test_scaled = scaler.transform(X_test[numeric_features]) # Scale only numeric features
```

5) inding the optimal value of ‘K’

```
acc = {}
for k in range(3, 30, 2):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)
    y_pred = knn.predict(X_test_scaled)
    acc[k] = accuracy_score(y_test, y_pred)

# PLotting K v/s accuracy graph
plt.plot(range(3,30,2), acc.values())
plt.xlabel('K')
plt.ylabel('Accuracy')
plt.show()
```



6) Training the model

```
knn = KNeighborsClassifier(n_neighbors=13)
knn.fit(X_train_scaled, y_train)
y_pred = knn.predict(X_test_scaled)
print(accuracy_score(y_test, y_pred))
```

```
0.16683333333333333
```