CODE:

```
import pandas as pd
# Load the dataset
df = pd.read_csv('train.csv')
# Display the first few rows to inspect the data
df.head()
import os
os.listdir()


import pandas as pd
# Try 'train.csv' if that's the name, or just 'train' if it has no extension
df = pd.read_csv('train.csv', nrows=100000)
# Check the top rows to confirm it loaded
df.head()
df.drop('id', axis=1, inplace=True)
from sklearn.preprocessing import LabelEncoder


label_encoders = {}
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])
        label_encoders[col] = le


from sklearn.model_selection import train_test_split
X = df.drop('click', axis=1)
y = df['click']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Use a smaller subset for KNN
X_train_small = X_train[:5000]
y_train_small = y_train[:5000]
```

```
X_test_small = X_test[:1000]
# Train KNN
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_small, y_train_small)
y_pred_knn = knn.predict(X_test_small)

acc_knn = accuracy_score(y_test[:1000], y_pred_knn)
print("KNN Accuracy (Sampled):", acc_knn)

# Plot
import matplotlib.pyplot as plt
plt.bar(['KNN (Sampled)'], [acc_knn])
plt.ylabel('Accuracy')
plt.title('KNN Accuracy on Sampled Data')
plt.ylim(0, 1)
plt.show()

import numpy as np
import matplotlib.pyplot as plt
# Simulate MAB using CTR (click) per ad (use 'banner_pos' as a proxy ad ID)
n_ads = df['banner_pos'].nunique()
n_rounds = 10000
ad_selected = []
clicks = df['click'].values[:n_rounds]
ads = df['banner_pos'].values[:n_rounds]
rewards = [0] * n_ads
counts = [0] * n_ads
```

```
for i in range(n_rounds):
    ad = np.random.randint(0, n_ads)
    if np.random.rand() < 0.1:  # epsilon-greedy exploration
        ad = np.random.randint(0, n_ads)
    else:
        ad = np.argmax(rewards)

    ad_selected.append(ad)
    reward = 1 if (ads[i] == ad and clicks[i] == 1) else 0
    counts[ad] += 1
    rewards[ad] += reward

# Plot
plt.bar(range(n_ads), rewards)
plt.xlabel('Ad (banner_pos)')
plt.ylabel('Total Clicks (Reward)')
plt.title('Multi-Armed Bandit - Ad Performance')
plt.show()

from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Scale data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Create ANN
model = Sequential()
model.add(Dense(64, input_dim=X.shape[1], activation='relu'))
```

```
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_scaled[:50000], y[:50000], epochs=5, batch_size=128)

# Evaluate
loss, acc = model.evaluate(X_scaled[50000:60000], y[50000:60000])
print("ANN Accuracy:", acc)

# Plot
plt.bar(['ANN'], [acc])
plt.ylim(0, 1)
plt.ylabel('Accuracy')
plt.title('ANN Accuracy')
plt.show()

import matplotlib.pyplot as plt
# Example accuracy values, replace with your actual results
acc_knn = 0.65        # Replace with your KNN accuracy
acc_bandit = 0.55     # Replace with your Bandit accuracy (e.g., reward rate)

# Create the comparison graph
methods = ['KNN', 'ANN', 'Multi-Armed Bandit']
accuracies = [acc_knn, acc_ann, acc_bandit]

# Plot
plt.figure(figsize=(8, 5))
bars = plt.bar(methods, accuracies, color=['blue', 'green', 'orange'])
plt.ylim(0, 1)
plt.ylabel('Accuracy / Average Reward')
```

```
plt.title('Model Comparison on Avazu CTR Dataset')


# Add value labels on top
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2.0, yval + 0.01, f'{yval:.2f}', ha='center', va='bottom')


plt.show()
```