```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
```

```
# Load dataset
df = pd.read_csv("/content/adult 3 (1).csv")  # Rename your file accordingly

# Drop rows with missing values
df.replace('?', np.nan, inplace=True)
df.dropna(inplace=True)

# Drop unnecessary columns
df.drop(['education'], axis=1, inplace=True)

# Label encode categorical features
le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = le.fit_transform(df[col])
```

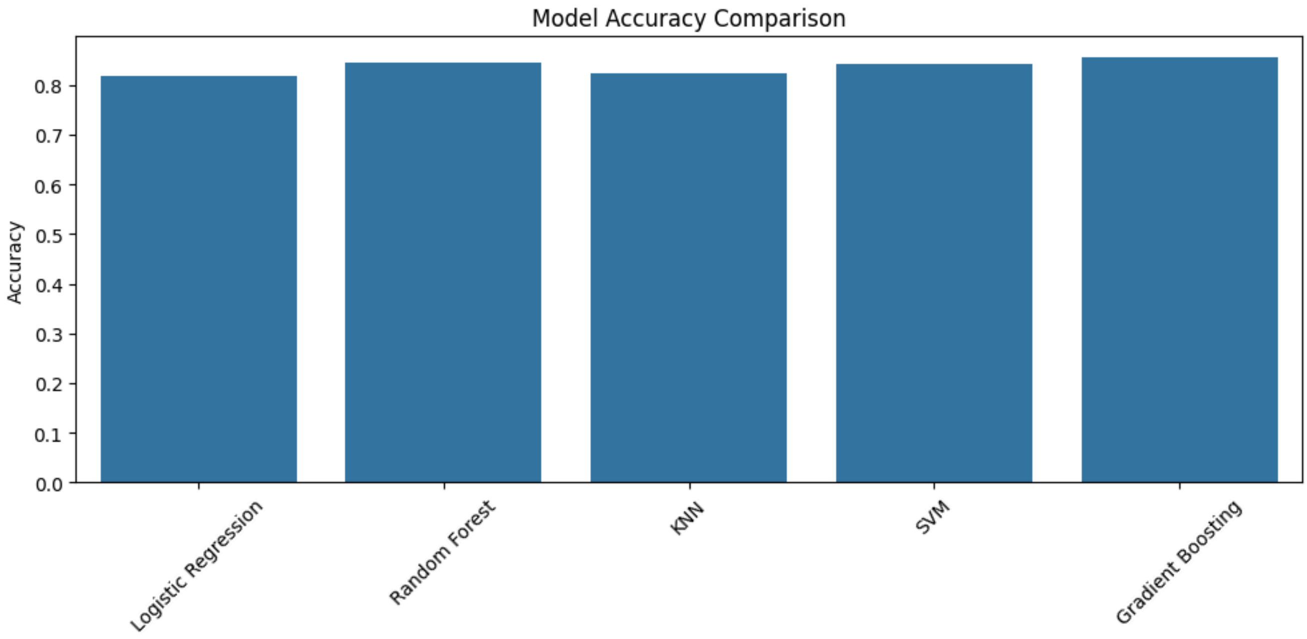
```
X = df.drop("income", axis=1)
y = df["income"]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(),
    "Gradient Boosting": GradientBoostingClassifier()
}

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    results[name] = acc
    print(f"{name}: {acc:.4f}")
    print(classification_report(y_test, y_pred))
```

```
Logistic Regression: 0.8194
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      2668
           1       0.72      0.46      0.56       887

    accuracy                           0.82      3555
   macro avg       0.78      0.70      0.72      3555
weighted avg       0.81      0.82      0.80      3555

Random Forest: 0.8459
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      2668
           1       0.73      0.61      0.66       887

    accuracy                           0.85      3555
   macro avg       0.80      0.77      0.78      3555
weighted avg       0.84      0.85      0.84      3555
```

```
KNN: 0.8233
              precision    recall  f1-score   support

           0       0.87      0.90      0.88      2668
           1       0.66      0.59      0.62       887

    accuracy                           0.82      3555
   macro avg       0.77      0.75      0.75      3555
weighted avg       0.82      0.82      0.82      3555

SVM: 0.8411
              precision    recall  f1-score   support

           0       0.86      0.94      0.90      2668
           1       0.75      0.54      0.63       887

    accuracy                           0.84      3555
   macro avg       0.81      0.74      0.76      3555
weighted avg       0.83      0.84      0.83      3555

Gradient Boosting: 0.8551
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      2668
           1       0.77      0.60      0.68       887

    accuracy                           0.86      3555
   macro avg       0.82      0.77      0.79      3555
weighted avg       0.85      0.86      0.85      3555
```

```
plt.figure(figsize=(10, 5))
sns.barplot(x=list(results.keys()), y=list(results.values()))
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
best_model_name = max(results, key=results.get)
best_model = models[best_model_name]
joblib.dump(best_model, "best_model.pkl")
print(f"Best model: {best_model_name} saved.")
```

```
Best model: Gradient Boosting saved.
```

```
from google.colab import files
files.download('best_model.pkl')
```