Name: Dhruvil Shah

Sap id: 60009210075

Batch: D12

Subject: ADS Lab 7

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pandas as pd

df = pd.read_csv('/content/car_evaluation.csv')
df.head()
```

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

```
df.shape
```

```
(1727, 7)
```

```
df.describe()
```

	vhigh	vhigh.1	2	2.1	small	low	unacc
count	1727	1727	1727	1727	1727	1727	1727
unique	4	4	4	3	3	3	4
top	high	high	3	4	med	med	unacc
freq	432	432	432	576	576	576	1209

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   vhigh    1727 non-null   object
 1   vhigh.1  1727 non-null   object
 2   2        1727 non-null   object
 3   2.1      1727 non-null   object
 4   small    1727 non-null   object
 5   low      1727 non-null   object
 6   unacc    1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

df.columns = col_names

col_names
```

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

```
df.isnull().sum()
```

```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

```
df['class'].value_counts()
```

```
unacc    1209
acc       384
good       69
vgood      65
Name: class, dtype: int64
```

```
X=df.drop(['class'],axis=1)
```

```
y=df['class']
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
encoder = OneHotEncoder(sparse=False, drop='first')
categorical_columns = X.select_dtypes(include=['object']).columns
X_encoded = encoder.fit_transform(X[categorical_columns])
column_names = encoder.get_feature_names_out(categorical_columns)
X_encoded = pd.DataFrame(X_encoded, columns=column_names)
X = pd.concat([X.drop(columns=categorical_columns), X_encoded], axis=1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define your machine learning model
model = RandomForestClassifier()

# Define a grid of hyperparameters to search
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create GridSearchCV object
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters from the grid search
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Train the model with the best hyperparameters on the entire training set
best_model = RandomForestClassifier(**best_params)
best_model.fit(X_train, y_train)

# Evaluate the model's performance on the test set
accuracy = best_model.score(X_test, y_test)
print("Model Accuracy:", accuracy)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outpu
  warnings.warn(
Best Hyperparameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Model Accuracy: 0.8815028901734104
```

```
import pickle
```

```
model = RandomForestClassifier(n_estimators=100, max_depth=10)
model.fit(X, y)

# Save the trained model as a pickle file
with open('trained_model.pkl', 'wb') as model_file:
    pickle.dump(model, model_file)
```