Machine Learning Project 1

Scikit-Learn and Cross Validation Project

Lakshmi Chakradhar Vijayarao

Net ID: lxv230000

Introduction

In this project, I built classification models to check if the mushroom is edible or poisonous using the mushroom_mixed_50000.csv dataset. I used three machine learning algorithms: Decision Tree Classifier, Random Forest Classifier and KNN Classifier. The models were evaluated using grid search and cross-validation, and the final model was saved for accuracy competition.

Dataset

The dataset contains 50,000 samples with 20 features. The target variable has two classes: e (edible mushroom) and p (poisonous mushroom).

Model Building

I used the scikit-learn library to implement two models:

- Decision Tree Classifier
- Random Forest Classifier
- KNN Classifier

Cross-Validation and Grid Search

I performed grid search with 5-fold cross-validation. The parameter grids for each model were as follows:

Model	Parameter	Values
Decision Tree	max depth	[10, 20, 30]
	min samples split	[2, 5, 10]
	min samples leaf	[1, 2, 4]
Random Forest	n estimators	[50, 100]
	max depth	[10, 15]
	min samples split	[5, 10]
	min samples leaf	[1, 2, 4]
	max features	['sqrt', 'log2']
KNN	neighbors	[3. 5. 7]
	Weights.	['uniform' , 'distance']

Table 1: Grid search parameters for Decision Tree, Random Forest and KNN

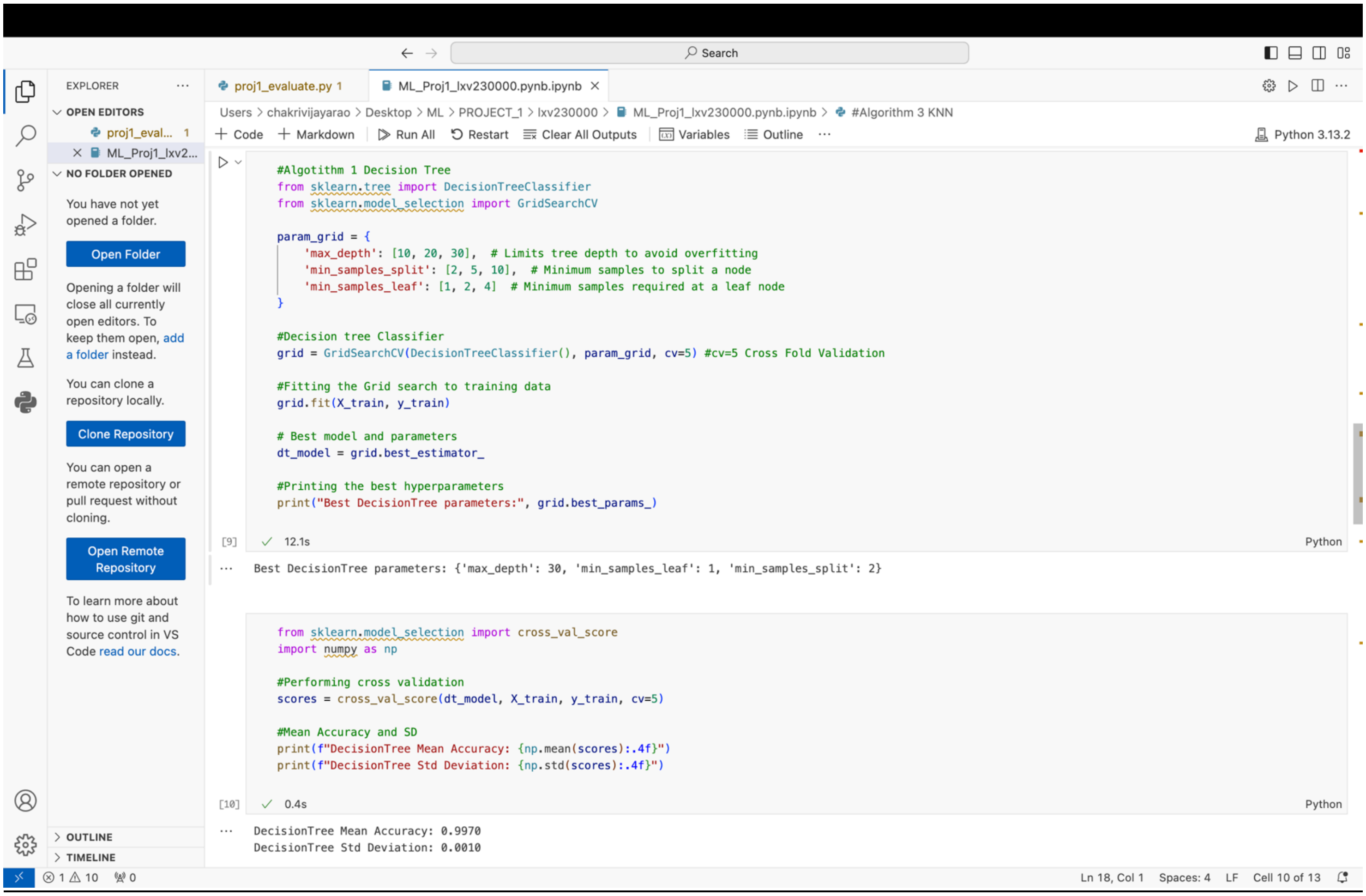
Documentation

The approach for this project can be summarized in the following steps:

- **Data Loading**: I began by loading the dataset using pandas. The dataset was read into a DataFrame, where the features (X) were separated from the target variable (y).
- **Train-Test Split**: I split the data into training and testing sets using train test split from scikit-learn. This ensured that our model would be evaluated on unseen data, providing a fair assessment of its performance.
- **Model Definition**: I defined three classifiers: DecisionTreeClassifier, RandomForestClassifier and KNNClassifier.
- **Parameter Tuning**: To find the best hyperparameters, we employed Grid Search with cross-validation. This involved defining a parameter grid for each model, which allowed me to systematically explore different combinations of parameters to find the optimal settings for our classifiers.
- **Model Evaluation**: After fitting the models with the training data, we evaluated their performance using accuracy scores calculated on the test data. I also recorded the mean accuracy and standard deviation from the cross-validation results.
- **Model Saving**: The best performing model based on accuracy was saved for future use. I utilized Python's pickle module for serialization.

Results

Decision Tree:



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

You can open a remote repository or pull request without cloning.

To learn more about how to use git and source control in VS Code read our docs.

```
#Algotithm 1 Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

param_grid = {
    'max_depth': [10, 20, 30],  # Limits tree depth to avoid overfitting
    'min_samples_split': [2, 5, 10],  # Minimum samples to split a node
    'min_samples_leaf': [1, 2, 4]  # Minimum samples required at a leaf node
}

#Decision tree Classifier
grid = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5) #cv=5 Cross Fold Validation

#Fitting the Grid search to training data
grid.fit(X_train, y_train)

# Best model and parameters
dt_model = grid.best_estimator_

#Printing the best hyperparameters
print("Best DecisionTree parameters:", grid.best_params_)
```

[9] ✓ 12.1s

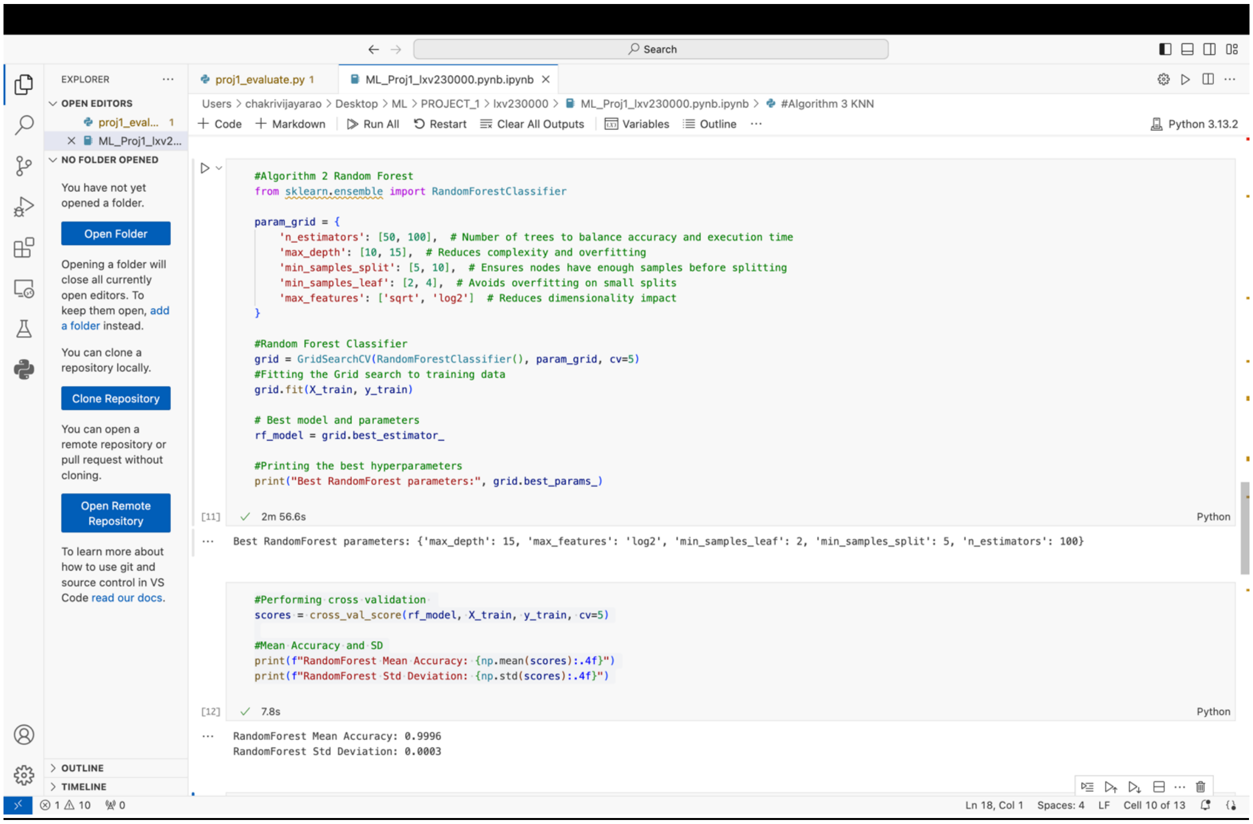
```
Best DecisionTree parameters: {'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

```
from sklearn.model_selection import cross_val_score
import numpy as np

#Performing cross validation
scores = cross_val_score(dt_model, X_train, y_train, cv=5)

#Mean Accuracy and SD
print(f"DecisionTree Mean Accuracy: {np.mean(scores):.4f}")
print(f"DecisionTree Std Deviation: {np.std(scores):.4f}")
```

[10] ✓ 0.4s

```
DecisionTree Mean Accuracy: 0.9970
DecisionTree Std Deviation: 0.0010
```

<u>Random Forest:</u>


proj1_evaluate.py 1
ML_Proj1_lxv230000.pynb.ipynb
Users > chakrivijayarao > Desktop > ML > PROJECT_1 > lxv230000 > ML_Proj1_lxv230000.pynb.ipynb > #Algorithm 3 KNN
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.13.2
Ln 18, Col 1 Spaces: 4 LF Cell 10 of 13

```
#Algorithm 2 Random Forest
from sklearn.ensemble import RandomForestClassifier

param_grid = {
    'n_estimators': [50, 100],  # Number of trees to balance accuracy and execution time
    'max_depth': [10, 15],  # Reduces complexity and overfitting
    'min_samples_split': [5, 10],  # Ensures nodes have enough samples before splitting
    'min_samples_leaf': [2, 4],  # Avoids overfitting on small splits
    'max_features': ['sqrt', 'log2']  # Reduces dimensionality impact
}

#Random Forest Classifier
grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
#Fitting the Grid search to training data
grid.fit(X_train, y_train)

# Best model and parameters
rf_model = grid.best_estimator_

#Printing the best hyperparameters
print("Best RandomForest parameters:", grid.best_params_)
```

[11] ✓ 2m 56.6s

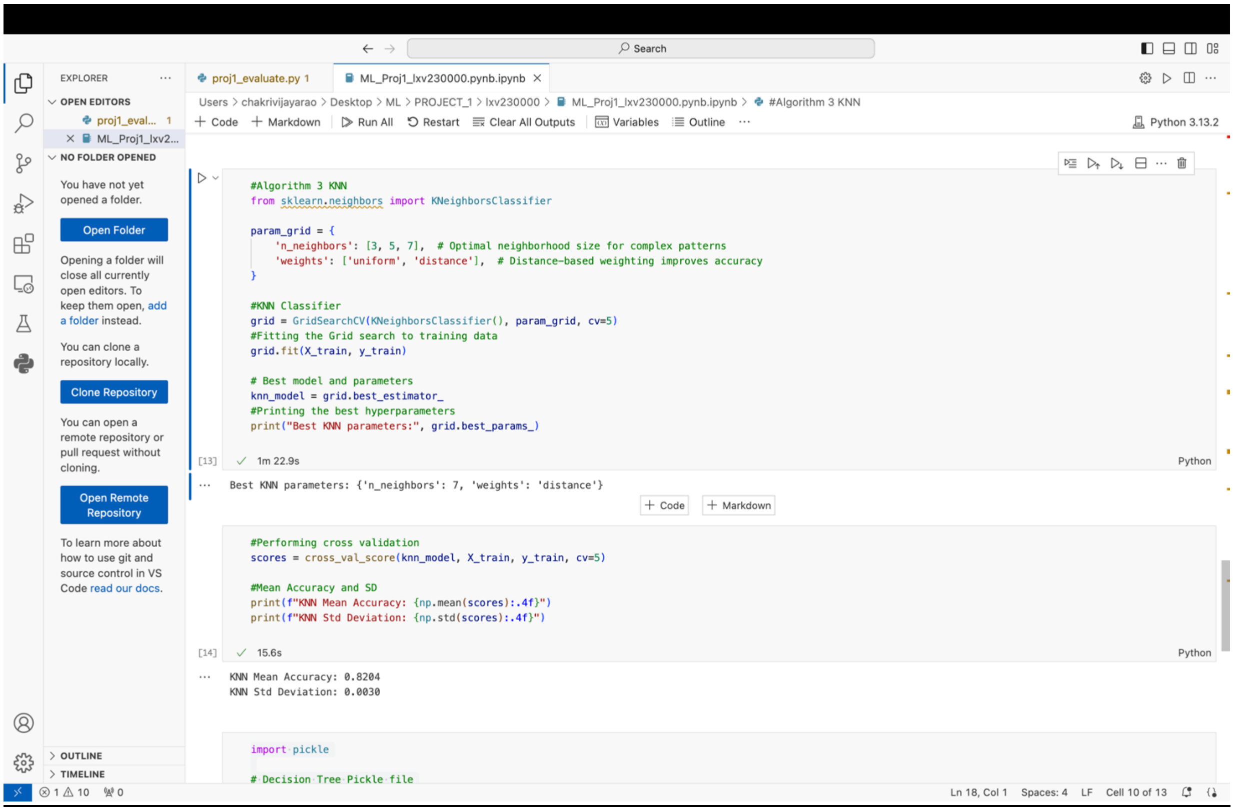
```
Best RandomForest parameters: {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}
```

```
#Performing cross validation
scores = cross_val_score(rf_model, X_train, y_train, cv=5)

#Mean Accuracy and SD
print(f"RandomForest Mean Accuracy: {np.mean(scores):.4f}")
print(f"RandomForest Std Deviation: {np.std(scores):.4f}")
```

[12] ✓ 7.8s

```
RandomForest Mean Accuracy: 0.9996
RandomForest Std Deviation: 0.0003
```

<u>KNN:</u>


proj1_evaluate.py 1
ML_Proj1_lxv230000.pynb.ipynb
Users > chakrivijayarao > Desktop > ML > PROJECT_1 > lxv230000 > ML_Proj1_lxv230000.pynb.ipynb > #Algorithm 3 KNN
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.13.2
Ln 18, Col 1 Spaces: 4 LF Cell 10 of 13

```
#Algorithm 3 KNN
from sklearn.neighbors import KNeighborsClassifier

param_grid = {
    'n_neighbors': [3, 5, 7],  # Optimal neighborhood size for complex patterns
    'weights': ['uniform', 'distance'],  # Distance-based weighting improves accuracy
}

#KNN Classifier
grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
#Fitting the Grid search to training data
grid.fit(X_train, y_train)

# Best model and parameters
knn_model = grid.best_estimator_
#Printing the best hyperparameters
print("Best KNN parameters:", grid.best_params_)
```

[13] ✓ 1m 22.9s

```
Best KNN parameters: {'n_neighbors': 7, 'weights': 'distance'}
```

```
#Performing cross validation
scores = cross_val_score(knn_model, X_train, y_train, cv=5)

#Mean Accuracy and SD
print(f"KNN Mean Accuracy: {np.mean(scores):.4f}")
print(f"KNN Std Deviation: {np.std(scores):.4f}")
```

[14] ✓ 15.6s

```
KNN Mean Accuracy: 0.8204
KNN Std Deviation: 0.0030
```

```
import pickle

# Decision Tree Pickle file
```

The results of grid search with cross-validation are summarized below:

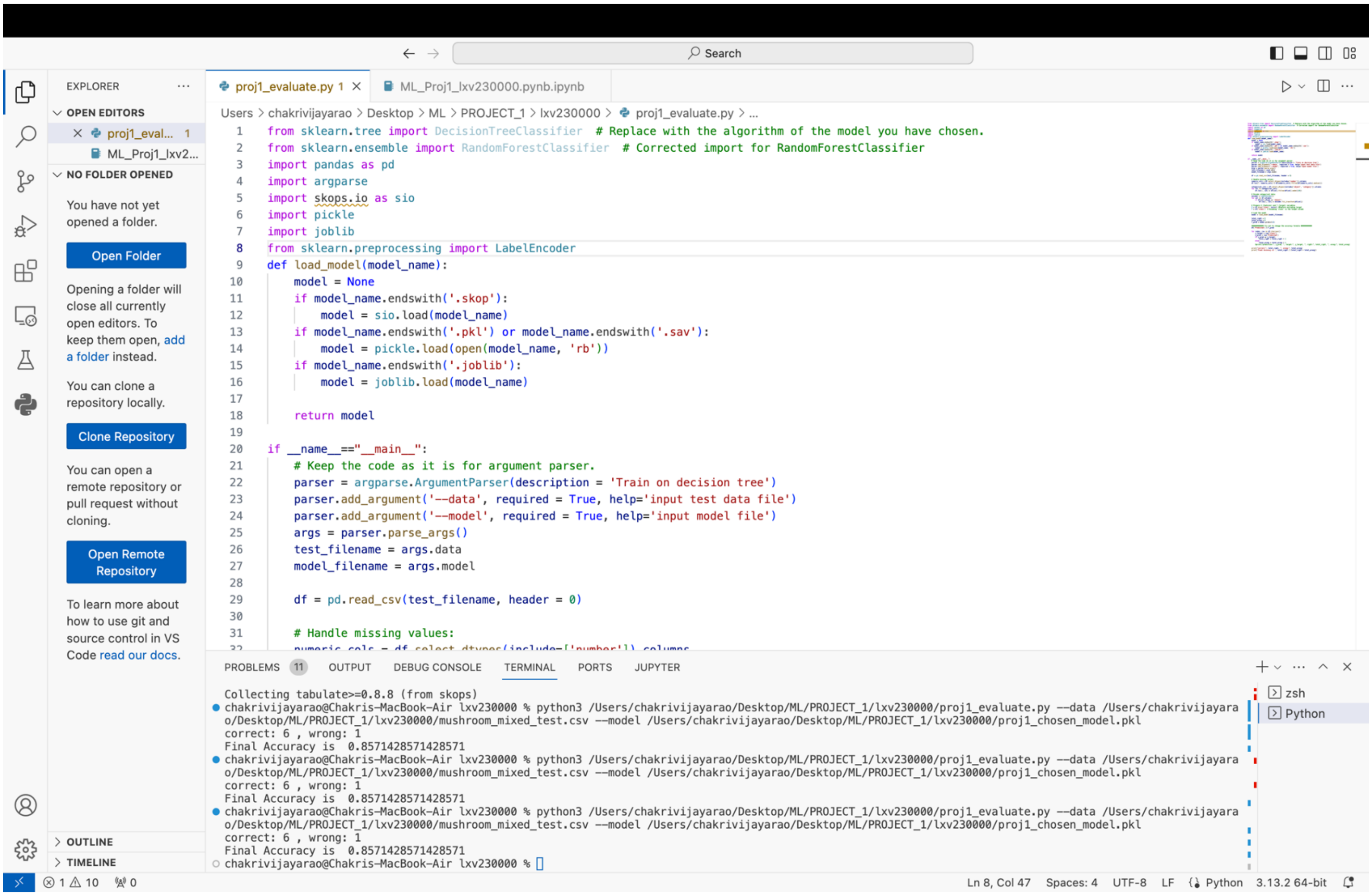
Hyperparameter	Decision Tree Classifier	Random Forest Classifier	KNN Classifier
max depth	30	15	-
max features	-	Log2	-
min samples leaf	1	2	-
min samples split	2	5	-
n estimators	-	100	-
N neighbours	-	-	7
weights	-	-	distance

Table 2: Best Parameters Found for Decision Tree and Random Forest Classifiers

Model	Mean Accuracy	Accuracy Std. Dev.
Decision Tree	0.9970	0.0010
Random Forest	0.9996	0.0003
KNN	0.8204	0.0030

Table 3: Cross-Validation and Test Accuracy Results

Evalvuation Program Output:


Search
EXPLORER
OPEN EDITORS
proj1_eval... 1
ML_Proj1_lxv2...
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can clone a repository locally.
Clone Repository
You can open a remote repository or pull request without cloning.
Open Remote Repository
To learn more about how to use git and source control in VS Code read our docs.
OUTLINE
TIMELINE
proj1_evaluate.py 1
ML_Proj1_lxv230000.pynb.ipynb
Users > chakrivijayarao > Desktop > ML > PROJECT_1 > lxv230000 > proj1_evaluate.py > ...
PROBLEMS 11
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
JUPYTER
zsh
Python
⊗ 1 ⚠ 10
0
Ln 8, Col 47
Spaces: 4
UTF-8
LF
Python
3.13.2 64-bit

```python
from sklearn.tree import DecisionTreeClassifier  # Replace with the algorithm of the model you have chosen.
from sklearn.ensemble import RandomForestClassifier  # Corrected import for RandomForestClassifier
import pandas as pd
import argparse
import skops.io as sio
import pickle
import joblib
from sklearn.preprocessing import LabelEncoder
def load_model(model_name):
    model = None
    if model_name.endswith('.skop'):
        model = sio.load(model_name)
    if model_name.endswith('.pkl') or model_name.endswith('.sav'):
        model = pickle.load(open(model_name, 'rb'))
    if model_name.endswith('.joblib'):
        model = joblib.load(model_name)

    return model

if __name__=="__main__":
    # Keep the code as it is for argument parser.
    parser = argparse.ArgumentParser(description = 'Train on decision tree')
    parser.add_argument('--data', required = True, help='input test data file')
    parser.add_argument('--model', required = True, help='input model file')
    args = parser.parse_args()
    test_filename = args.data
    model_filename = args.model

    df = pd.read_csv(test_filename, header = 0)

    # Handle missing values:
    numeric_cols = df.select_dtypes(include=['number']).columns
```

```
Collecting tabulate>=0.8.8 (from skops)
chakrivijayarao@Chakris-MacBook-Air lxv230000 % python3 /Users/chakrivijayarao/Desktop/ML/PROJECT_1/lxv230000/proj1_evaluate.py --data /Users/chakrivijayara
o/Desktop/ML/PROJECT_1/lxv230000/mushroom_mixed_test.csv --model /Users/chakrivijayarao/Desktop/ML/PROJECT_1/lxv230000/proj1_chosen_model.pkl
correct: 6 , wrong: 1
Final Accuracy is  0.8571428571428571
chakrivijayarao@Chakris-MacBook-Air lxv230000 % python3 /Users/chakrivijayarao/Desktop/ML/PROJECT_1/lxv230000/proj1_evaluate.py --data /Users/chakrivijayara
o/Desktop/ML/PROJECT_1/lxv230000/mushroom_mixed_test.csv --model /Users/chakrivijayarao/Desktop/ML/PROJECT_1/lxv230000/proj1_chosen_model.pkl
correct: 6 , wrong: 1
Final Accuracy is  0.8571428571428571
chakrivijayarao@Chakris-MacBook-Air lxv230000 % python3 /Users/chakrivijayarao/Desktop/ML/PROJECT_1/lxv230000/proj1_evaluate.py --data /Users/chakrivijayara
o/Desktop/ML/PROJECT_1/lxv230000/mushroom_mixed_test.csv --model /Users/chakrivijayarao/Desktop/ML/PROJECT_1/lxv230000/proj1_chosen_model.pkl
correct: 6 , wrong: 1
Final Accuracy is  0.8571428571428571
chakrivijayarao@Chakris-MacBook-Air lxv230000 %
```

Conclusion

The Random Forest model achieved the highest accuracy of 99.96%, and it was saved as proj1_chosen_model.pkl.