Project 1: Random Forests

Overview

Theory: Overleaf Link

This project consists of two file categories: `demos` and `run.py` and `model.py`. These files are designed to perform a specific task or set of tasks related to a machine learning model or application. Below is a brief description of each file:

1 Main Folder

`demo_basics.py`

Main header. Contains classes and functions.

2 to_demo

`demo_all_gains.py`

`demo_drop_and_split.py`

`demo_impute_and_split.py`

`demo_one_tree.py`

`demo_tree_graph.py`

`gains.py`

Displays gains for all features in train.csv.

`continuous_to_categorical.py`

Utility to convert continuous data to categorical.

chi_squared_signf.py

Used to calculate chi-squared and determine significance.

3 Main Folder/for_kaggle

demo_70%_onetree_nodrop_undersample.py

demo_card1.one-tree.no_drop.py

run.py

run.py

run.py

run.py

run.py

run.py

This file serves as the entry point for running the machine learning model or application. It typically contains code for setting up the environment, loading data, training the model, and evaluating its performance. Users can execute this file to start the application or train the model.

Usage

To run the application or train the model, execute the following command:

```
python run.py
```

model.py

The `model.py` file contains the implementation of the machine learning model used in the project. It includes classes and functions related to data preprocessing, model architecture, training, evaluation, and inference. This file encapsulates the core functionality of the model and provides an interface for interacting with it.

Usage

To fit our model, use *DecisionNode.fit*:

```
import demo_basics as demo

# load data
df =  pd.read_csv(os.getcwd() + '/train.csv')

# build forest
forest = []
for i in demo.categorical:
    one_tree = demo.DecisionNode()
    one_tree.fit(df[i], df["isFraud"])
    forest.append(one_tree)
```

To predict with our model, use *DecisionNode.predict*:

```
none_returned = 0
predictions = []
for (_,row) in X_test.iterrows():

    predict_of = []
    for idx,column in enumerate(demo.categorical):
        value = row[column]
        chance_of_1 = one_tree.predict(value)



        if chance_of_1 == None:
            none_returned += 1
            predict_of.append(np.random.randint(2))    # not found
        elif chance_of_1 < 0.5:
            predict_of.append(0)
        else:
            predict_of.append(1)
    vote = sum(predict_of)/len(demo.categorical)
    if vote < 0.5:
        predictions.append(0)
    else:
        predictions.append(1)

timer_end = time.time()
print(f"time to predict with one-tree node: {timer_end-timer_start}")
```

Kaggle Submissions

```
p_nones = none_returned/len(X_test)
print("none_returned=%.2f" % p_nones)
p_found = (1.0-p_nones)
print("found_accuracy=%.2f" % p_found)


if 0:
    % for kaggle submission
    start_at = 472433
    out_index = range(start_at, start_at+len(df_test))
    out_predictions = predictions
    out_pd = pd.DataFrame({"TransactionID":out_index, "isFraud":out_predictions})
    out_pd = out_pd.set_index("TransactionID", drop=True)
    out_pd.to_csv(os.getcwd() + "/out_forest.csv")
end
```

Requirements

Ensure you have the following dependencies installed to run the project:

- Python (anaconda) version 3.11.5
- Additional libraries:(NumPy, Pandas, Scikit-learn) specified in `requirements.txt`

License

This project is licensed under the MIT License.