The Superior University, Lahore

Assignment-I (Fall 2024)

Course Title:	PAI				Course Code:	CPR601260	Credit Hours:	4
Instructor:					Programme Name:	BSDS		
Semester:	4th	Batch:	F23	Section:	BSDSM-4A	Date:	30 January, 2024	
Time Allowed:					Maximum Marks:			
Student's Name:	Fizza Farooq				Reg. No.	Su92-bsdsm-f23-017		
Lab-Task 1: Question 2: Question								

TASK 2

Competition of Space titanic on kaggle

Accuracy Score

Code and its output with explanation

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/spaceship-titanic/sample_submission.csv
/kaggle/input/spaceship-titanic/train.csv
/kaggle/input/spaceship-titanic/test.csv
```

Importing Libraries and Checking for Data Files

Imports `numpy` and `pandas` for data processing.

Uses `os.walk()` to list all files in the `/kaggle/input` directory to confirm that the dataset is available.

```
import tensorflow as tf
import tensorflow_decision_forests as tfdf
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
print("TensorFlow v" + tf.__version__)
print("TensorFlow Decision Forests v" + tfdf.__version__)
```

```
TensorFlow v2.17.1
TensorFlow Decision Forests v1.10.0
```

Additional Library Imports

- Imports TensorFlow for machine learning.
- Imports tensorflow_decision_forests (TFDF) for decision tree-based models.
- Imports seaborn and matplotlib.pyplot for data visualization.
- Prints the versions of TensorFlow and TFDF.

```
df=pd.read_csv("/kaggle/input/spaceship-titanic/train.csv")
df
```

- Reads the dataset train.csv into a Pandas DataFrame df.
- Displays the dataset in the notebook.

```
[7] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   PassengerId   8693 non-null   object
 1   HomePlanet    8492 non-null   object
 2   CryoSleep     8476 non-null   object
 3   Cabin         8494 non-null   object
 4   Destination   8511 non-null   object
 5   Age           8514 non-null   float64
 6   VIP           8490 non-null   object
 7   RoomService   8512 non-null   float64
 8   FoodCourt     8510 non-null   float64
 9   ShoppingMall  8485 non-null   float64
 10  Spa           8510 non-null   float64
 11  VRDeck        8505 non-null   float64
 12  Name          8493 non-null   object
 13  Transported   8693 non-null   bool
dtypes: bool(1), float64(6), object(7)
memory usage: 891.5+ KB
```

- Displays dataset structure, including column names, data types, and missing values.

```
df.count()
```

```
PassengerId     8693
HomePlanet      8492
CryoSleep       8476
Cabin           8494
Destination     8511
Age             8514
VIP             8490
RoomService     8512
FoodCourt       8510
ShoppingMall    8485
Spa             8510
VRDeck          8505
Name            8493
Transported     8693
dtype: int64
```

- Shows the count of non-null values in each column.

```
df = df.drop(['PassengerId', 'Name'], axis=1)
```

Removes Passenger Id and Name from the dataset since they are not useful for analysis.

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

```
<bound method DataFrame.sum of       HomePlanet  CryoSleep  Cabin  Destination    Age    VIP  RoomService
0          False      False  False        False  False  False        False
1          False      False  False        False  False  False        False
2          False      False  False        False  False  False        False
3          False      False  False        False  False  False        False
4          False      False  False        False  False  False        False
...          ...        ...    ...          ...    ...    ...          ...
8688       False      False  False        False  False  False        False
8689       False      False  False        False  False  False        False
8690       False      False  False        False  False  False        False
8691       False      False  False        False  False  False        False
8692       False      False  False        False  False  False        False

      FoodCourt  ShoppingMall    Spa  VRDeck  Transported
0         False         False  False   False        False
1         False         False  False   False        False
2         False         False  False   False        False
3         False         False  False   False        False
4         False         False  False   False        False
...         ...           ...    ...     ...          ...
8688      False         False  False   False        False
8689      False         False  False   False        False
8690      False         False  False   False        False
8691      False         False  False   False        False
8692      False         False  False   False        False

[8693 rows x 12 columns]>
```

Prints the function reference, **not the actual missing value count** (should be df.isnull().sum() instead).

```
df.isnull().sum().sort_values(ascending=False)
```

```
CryoSleep       217
ShoppingMall    208
VIP             203
HomePlanet      201
Cabin           199
VRDeck          188
FoodCourt       183
Spa             183
Destination     182
RoomService     181
Age             179
Transported       0
dtype: int64
```

- Corrected version that shows missing values in descending order.

```
df[['VIP', 'CryoSleep', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']] = df[['VIP', 'CryoSleep', 'FoodCourt', 'ShoppingMall',
                                                                            'Spa', 'VRDeck']].fillna(value=0)
df.isnull().sum().sort_values(ascending=False)
df
```

- Fills missing values in categorical and numerical columns with 0.

```
df[["Deck", "Cabin_num", "Side"]] = df["Cabin"].str.split("/", expand=True)
```

Converts Transported, VIP, and CryoSleep columns to integers (1 or 0) for model training

```
def split_dataset(dataset, test_ratio=0.20):
    test_indices = np.random.rand(len(dataset)) < test_ratio
    return dataset[~test_indices], dataset[test_indices]

train_ds_pd, valid_ds_pd = split_dataset(df)
print("{} examples in training, {} examples in testing.".format(
    len(train_ds_pd), len(valid_ds_pd)))
```

```
6967 examples in training, 1726 examples in testing.
```

- Defines a function split_dataset() that randomly splits the dataset into training (80%) and testing (20%) sets.
- Prints the number of examples in each set.

```
import tensorflow_decision_forests as tfdf
train_ds = tfdf.keras.pd_dataframe_to_tf_dataset(train_ds_pd, label=label)
valid_ds = tfdf.keras.pd_dataframe_to_tf_dataset(valid_ds_pd, label=label)
```

```
tfdf.keras.get_all_models()
```

```
[tensorflow_decision_forests.keras.RandomForestModel,
 tensorflow_decision_forests.keras.GradientBoostedTreesModel,
 tensorflow_decision_forests.keras.CartModel,
 tensorflow_decision_forests.keras.DistributedGradientBoostedTreesModel]
```

- Converts the Pandas DataFrame into a format suitable for TensorFlow Decision Forests.

```
rf = tfdf.keras.RandomForestModel()
rf.compile(metrics=["accuracy"])
```

```
Use /tmp/tmpbnrzjxy4 as temporary training directory
```

- Initializes a RandomForestModel from TensorFlow Decision Forests.
- Trains the model using the train_ds dataset.

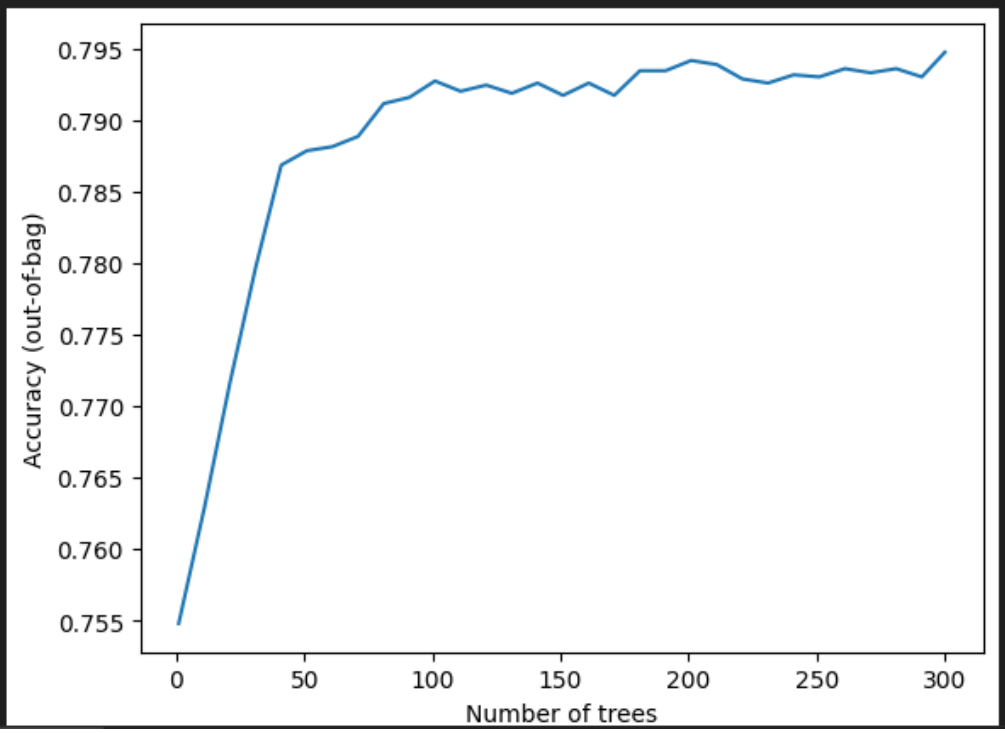
```
rf.fit(x=train_ds)
```

```
Reading training dataset...
Training dataset read in 0:00:05.190507. Found 6967 examples.
Training model...
Model trained in 0:00:53.511512
Compiling model...
Model compiled.

<tf_keras.src.callbacks.History at 0x7e5776beec20>
```

```
tfdf.model_plotter.plot_model_in_colab(rf, tree_idx=0, max_depth=3)
```

```
import matplotlib.pyplot as plt
logs = rf.make_inspector().training_logs()
plt.plot([log.num_trees for log in logs], [log.evaluation.accuracy for log in logs])
plt.xlabel("Number of trees")
plt.ylabel("Accuracy (out-of-bag)")
plt.show()
```




```
inspector = rf.make_inspector()
inspector.evaluation()
```

```
Evaluation(num_examples=6967, accuracy=0.7947466628390986, loss=0.5110323830903778, rmse=None, ndcg=None, aucs=None, auuc=None, qini=None)
```

```python
evaluation = rf.evaluate(x=valid_ds,return_dict=True)

for name, value in evaluation.items():
    print(f"{name}: {value:.4f}")
```

```
2/2 [==============================] - 1s 74ms/step - loss: 0.0000e+00 - accuracy: 0.7984
loss: 0.0000
accuracy: 0.7984
```

```python
print(f"Available variable importances:")
for importance in inspector.variable_importances().keys():
    print("\t", importance)
```

```
Available variable importances:
        INV_MEAN_MIN_DEPTH
        NUM_NODES
        SUM_SCORE
        NUM_AS_ROOT
```

```python
inspector.variable_importances()["NUM_AS_ROOT"]
```

```
[("CryoSleep" (1; #2), 109.0),
 ("RoomService" (1; #7), 68.0),
 ("Spa" (1; #10), 55.0),
 ("VRDeck" (1; #12), 28.0),
 ("ShoppingMall" (1; #8), 23.0),
 ("FoodCourt" (1; #5), 11.0),
 ("Deck" (4; #3), 4.0),
 ("HomePlanet" (4; #6), 2.0)]
```

```python
test_df = pd.read_csv('/kaggle/input/spaceship-titanic/test.csv')
submission_id = test_df.PassengerId

test_df[['VIP', 'CryoSleep']] = test_df[['VIP', 'CryoSleep']].fillna(value=0)

test_df[["Deck", "Cabin_num", "Side"]] = test_df["Cabin"].str.split("/", expand=True)
test_df = test_df.drop('Cabin', axis=1)

test_df['VIP'] = test_df['VIP'].astype(int)
test_df['CryoSleep'] = test_df['CryoSleep'].astype(int)

test_ds = tfdf.keras.pd_dataframe_to_tf_dataset(test_df)
predictions = rf.predict(test_ds)
n_predictions = (predictions > 0.5).astype(bool)
output = pd.DataFrame({'PassengerId': submission_id,
                       'Transported': n_predictions.squeeze()})

output.head()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:1458: RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:1459: RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
5/5 [==============================] - 1s 81ms/step
```

- Reads the **test dataset** from Kaggle.
- Extracts **PassengerId** for the final submission.
- **Fills missing values** in VIP and CryoSleep with 0 (assuming missing values mean "No").
- **Splits the Cabin column** into Deck, Cabin_num, and Side.
- Drops the original Cabin column.
- Converts VIP and CryoSleep to integers for consistency.
- Converts test_df into a TensorFlow dataset format.
- Uses the trained **Random Forest model** (rf) to predict.
- Converts prediction probabilities to boolean values (True/False) based on a 0.5 threshold.
- The test dataset might be missing columns that the model expects (especially categorical columns that were converted to numbers in training).
- You might need to **ensure that the test data has the same preprocessing as the training data.**

```python
sample_submission_df = pd.read_csv('/kaggle/input/spaceship-titanic/sample_submission.csv')
sample_submission_df['Transported'] = n_predictions
sample_submission_df.to_csv('/kaggle/working/submission.csv', index=False)
sample_submission_df.head()
```

- Reads **sample_submission.csv** (which contains the expected submission format).
- Replaces the Transported column with your predictions.
- Saves the final submission file as **submission.csv**.
- If n_predictions is not correctly formatted (e.g., incorrect shape or NaN values), it may cause submission errors.

- Check if n_predictions.shape matches sample_submission_df.shape.