SUPERIOR UNIVERSITY

Submitted By:

Name: **Um e Habiba**

Roll number **Su92-bsdsm-f23-040**

Section: **4A**

Task: **02**

Subject: **PAI (lab)**

Submitted to: **Sir Rasikh**

https://www.kaggle.com/code/umehabibaakbarali/spaceship-titanic

Introduction

The Spaceship Titanic dataset is a Kaggle competition dataset that involves predicting whether passengers were transported to another dimension. This report explains the structure of the Jupyter Notebook, the terms used, and its applications.

Overview of the Notebook

- Data loading and preprocessing
- Exploratory Data Analysis (EDA)
- Feature engineering
- Model training using machine learning techniques
- Predictions and evaluation

Key Terms and Their Definitions

1. **NumPy (numpy):** A library for numerical computations in Python, particularly useful for handling arrays and mathematical operations.
2. **Pandas (pandas):** A library for data manipulation, allowing users to read, modify, and analyze tabular data.
3. **Operating System Module (os):** A Python module used to interact with the operating system, such as reading files from directories.
4. **CSV (pd.read_csv):** A file format used to store tabular data, which can be loaded into a DataFrame using pandas.
5. **Exploratory Data Analysis (EDA):** A process used to understand the data through visualizations, statistical summaries, and data cleaning.
6. **Feature Engineering:** The process of creating new input features from existing data to improve model performance.
7. **Machine Learning (tensorflow, tensorflow_decision_forests):** Libraries used for training and implementing predictive models, including decision forests and neural networks.
8. **Classification Task:** A type of machine learning problem where the goal is to assign categories (e.g., predicting whether a passenger was transported or not).
9. **Model Training**: The process of teaching a machine learning model to recognize patterns in the dataset.

10. **Evaluation Metrics:** Methods used to measure the performance of the model, such as accuracy, precision, recall, and F1-score.

Code Explanation

1. Importing Libraries

Essential Python libraries such as NumPy, Pandas, OS, TensorFlow, and TensorFlow Decision Forests are imported to handle data processing, machine learning, and system operations.

2. Loading the Dataset

The dataset is loaded from the Kaggle environment using pd.read_csv(), allowing further analysis and processing.

3. Data Exploration and Preprocessing

- Checking missing values and handling them using techniques like imputation.
- Converting categorical features into numerical formats for machine learning models.
- Scaling numerical features for better model performance.

4. Feature Engineering

- Extracting meaningful features from existing data to enhance model predictions.
- Creating new columns based on domain knowledge.

5. Model Selection and Training

- Implementing machine learning models using TensorFlow and Decision Forests.
- Training models on the processed dataset.
- Tuning hyperparameters to improve model accuracy.

6. Model Evaluation

- Using classification metrics such as accuracy, precision, recall, and F1-score to assess the model's performance.
- Visualizing results using plots and confusion matrices.

7. Prediction Process

- The trained model is used to make predictions on new or test data.
- The test dataset is preprocessed in the same way as the training data to ensure consistency.
- The model's predict() function is applied to generate predictions.

- Predictions are typically stored in a new DataFrame and may be saved as a CSV file for submission.
- **Example:**
- predictions = model.predict(test_data)
- submission = pd.DataFrame({'PassengerId': test_data['PassengerId'], 'Transported': predictions})
- submission.to_csv('submission.csv', index=False)
- The output is used to determine the accuracy and effectiveness of the model.

Applications of the Notebook

- **Predictive Analytics:** Understanding trends and making forecasts based on historical data.
- **Data Cleaning & Preprocessing:** Handling missing values, encoding categorical data, and preparing data for modeling.
- **Model Deployment:** Using trained models in real-world applications such as customer segmentation and recommendation systems.
- **Feature Engineering:** Extracting meaningful information from raw datasets to improve prediction accuracy.
- **Deep Learning Implementation:** Applying advanced machine learning techniques to improve model accuracy.

Conclusion

The Spaceship Titanic notebook demonstrates a structured approach to solving a classification problem using machine learning. The notebook covers data processing, exploratory analysis, feature engineering, and predictive modeling using TensorFlow. Understanding these techniques is essential for data science projects involving structured datasets.

```
# 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
```

Libraries

```
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
```

```
dataset_df = pd.read_csv('/kaggle/input/spaceship-titanic/train.csv')
print("Full train dataset shape is {}".format(dataset_df.shape))
```

```
Full train dataset shape is (8693, 14)
```

```
dataset_df.head(5)
```

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Transported
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	Maham Ofracculy	False
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	Juanna Vines	True
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	Altark Susent	False
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	Solam Susent	False
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.0	2.0	Willy Santantines	True

[5]
```
dataset_df.describe()
```

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	38.000000	47.000000	76.000000	27.000000	59.000000	46.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

[6]
```
dataset_df.info()
```

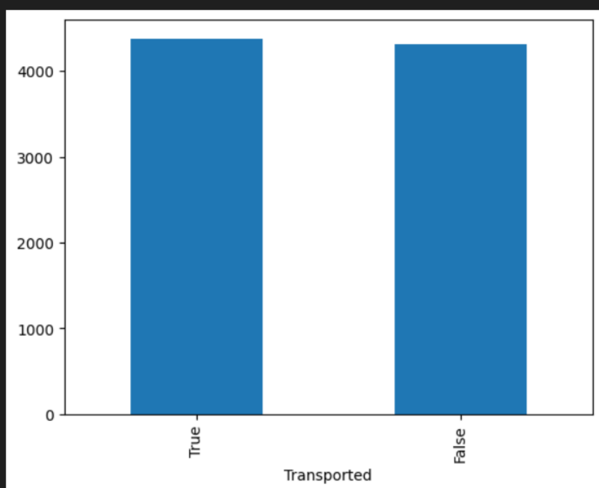
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   PassengerId   8693 non-null   object
 1   HomePlanet    8492 non-null   object
 2   CryoSleep     8476 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
```

```
 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
```

```python
plot_df = dataset_df.Transported.value_counts()
plot_df.plot(kind="bar")
```

[7]

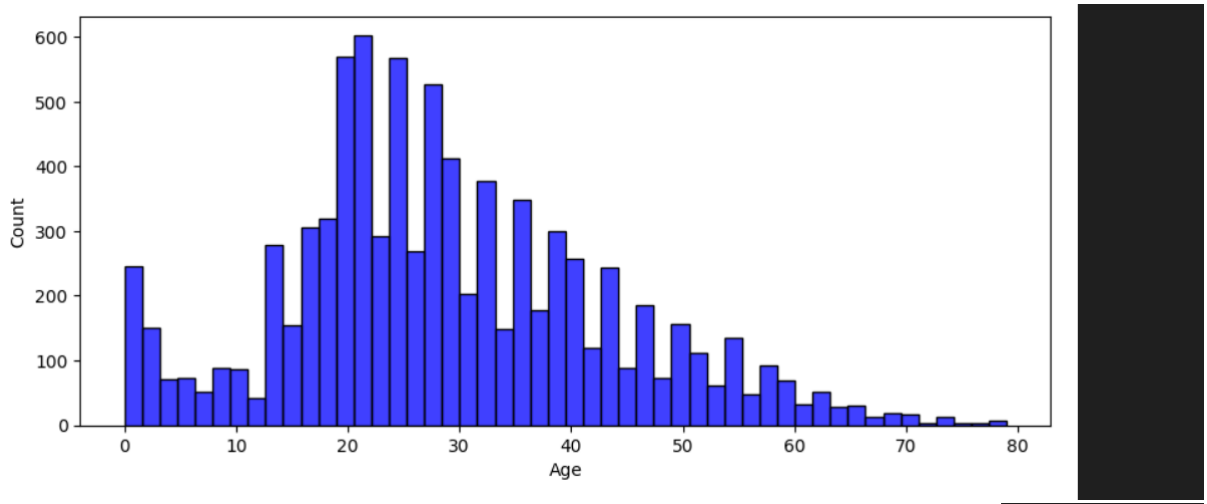
```
<Axes: xlabel='Transported'>
```

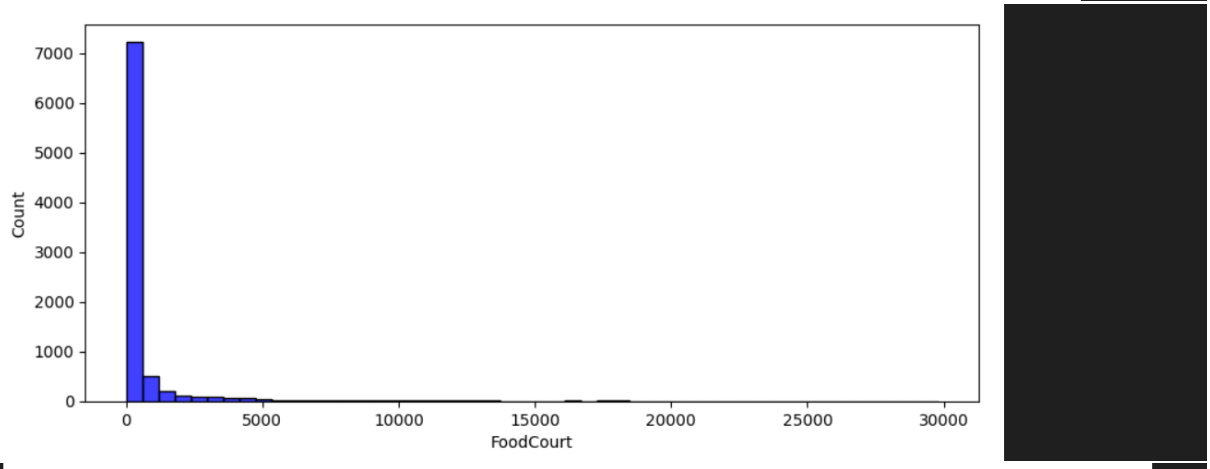


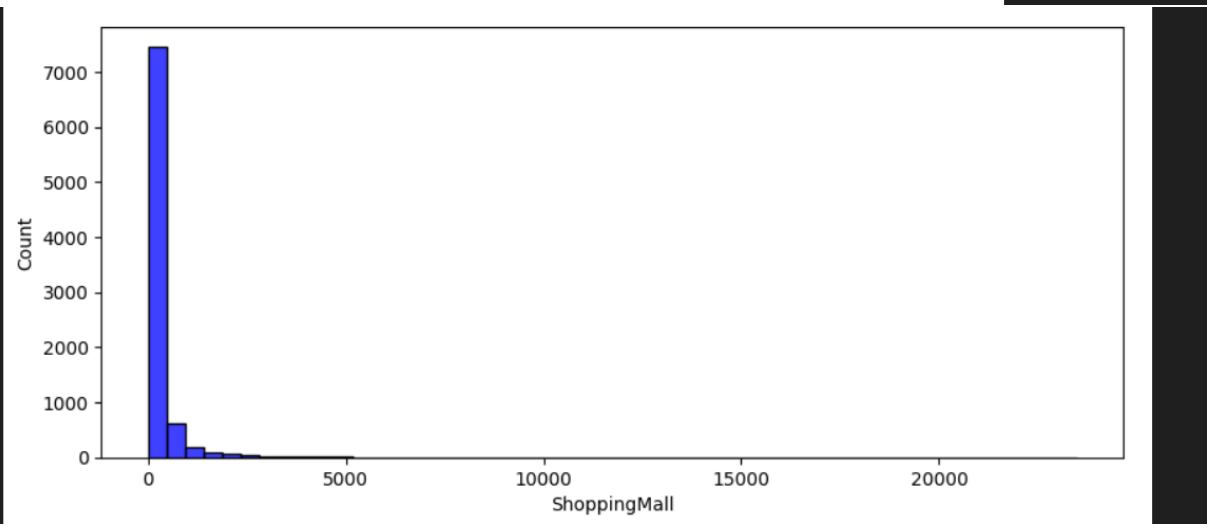

```python
fig, ax = plt.subplots(5,1,  figsize=(10, 10))
plt.subplots_adjust(top = 2)

sns.histplot(dataset_df['Age'], color='b', bins=50, ax=ax[0]);
sns.histplot(dataset_df['FoodCourt'], color='b', bins=50, ax=ax[1]);
sns.histplot(dataset_df['ShoppingMall'], color='b', bins=50, ax=ax[2]);
sns.histplot(dataset_df['Spa'], color='b', bins=50, ax=ax[3]);
sns.histplot(dataset_df['VRDeck'], color='b', bins=50, ax=ax[4]);
```

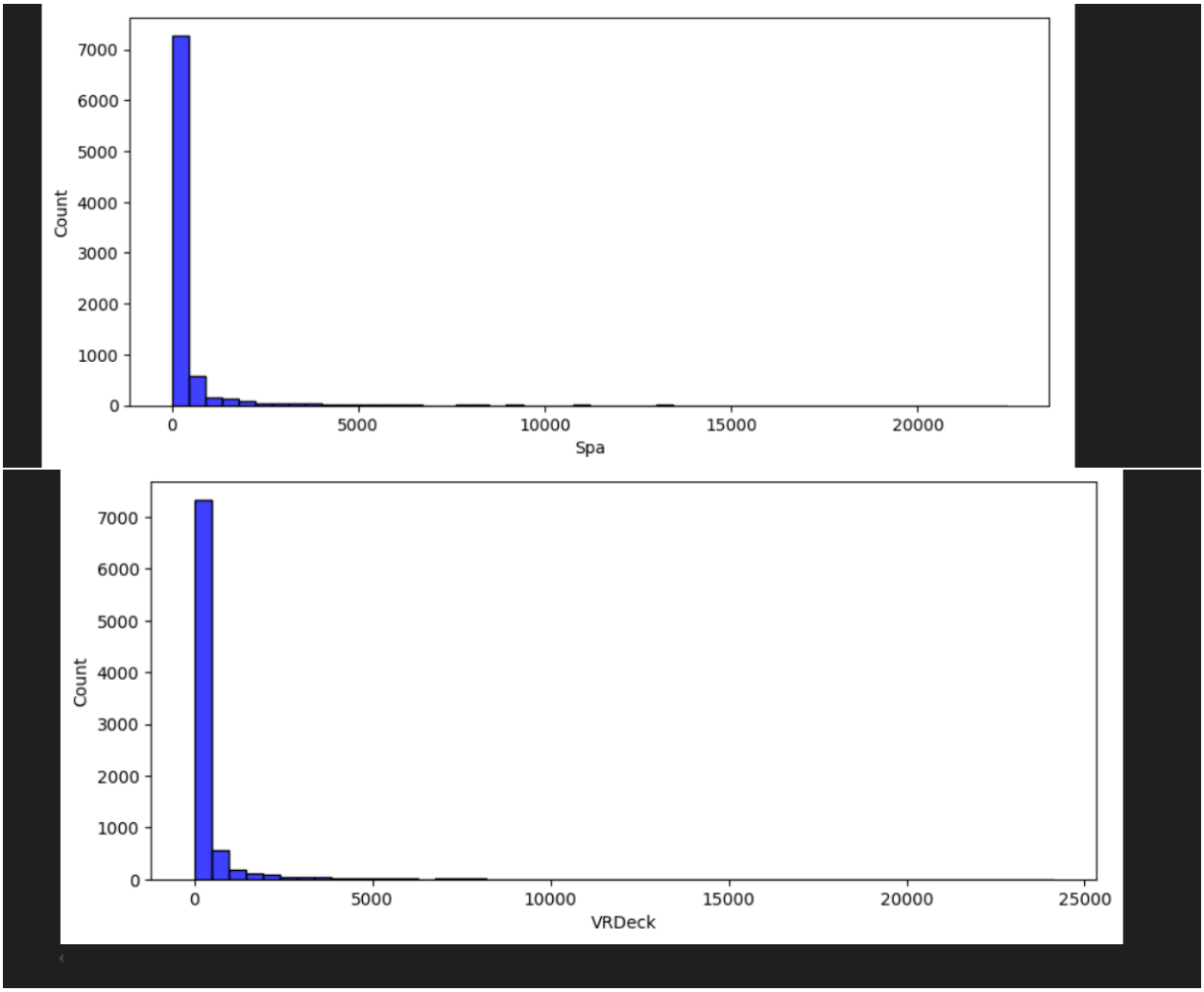
Python

```
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN be
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN be
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN be
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN be
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN be
  with pd.option_context('mode.use_inf_as_na', True):
```


Count
Age
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
80


Count
FoodCourt
0
1000
2000
3000
4000
5000
6000
7000
0
5000
10000
15000
20000
25000
30000


Count
ShoppingMall
0
1000
2000
3000
4000
5000
6000
7000
0
5000
10000
15000
20000


Count
7000
6000
5000
4000
3000
2000
1000
0
0
5000
10000
15000
20000
Spa
Count
7000
6000
5000
4000
3000
2000
1000
0
0
5000
10000
15000
20000
25000
VRDeck

```
[9]
dataset_df = dataset_df.drop(['PassengerId', 'Name'], axis=1)
dataset_df.head(5)
```

	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Transported
0	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	False
1	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	True
2	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	False
3	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	False
4	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.0	2.0	True

```
[10]
dataset_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
```

```
[11]
dataset_df[['VIP', 'CryoSleep', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']] = dataset_df[['VIP', 'CryoSleep', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']].fillna(value=0)
dataset_df.isnull().sum().sort_values(ascending=False)
```
Python

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

```
[12]
label = "Transported"
dataset_df[label] = dataset_df[label].astype(int)
```
Python

```
[13]
dataset_df['VIP'] = dataset_df['VIP'].astype(int)
dataset_df['CryoSleep'] = dataset_df['CryoSleep'].astype(int)
```
Python

```
[14]
dataset_df[["Deck", "Cabin_num", "Side"]] = dataset_df["Cabin"].str.split("/", expand=True)
```
Python

```
try:
    dataset_df = dataset_df.drop('Cabin', axis=1)
except KeyError:
    print("Field does not exist")
```

[15]

```
dataset_df.head(5)
```

[16]

	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Transported	Deck	Cabin_num	Side
0	Europa	0	TRAPPIST-1e	39.0	0	0.0	0.0	0.0	0.0	0.0	0	B	0	P
1	Earth	0	TRAPPIST-1e	24.0	0	109.0	9.0	25.0	549.0	44.0	1	F	0	S
2	Europa	0	TRAPPIST-1e	58.0	1	43.0	3576.0	0.0	6715.0	49.0	0	A	0	S
3	Europa	0	TRAPPIST-1e	33.0	0	0.0	1283.0	371.0	3329.0	193.0	0	A	0	S
4	Earth	0	TRAPPIST-1e	16.0	0	303.0	70.0	151.0	565.0	2.0	1	F	1	S

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

[17]

```
6963 examples in training, 1730 examples in testing.
```

```
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)
```

[18]

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

[19]

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

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

[20]

```
Use /tmp/tmpcsd_m1o9 as temporary training directory
```

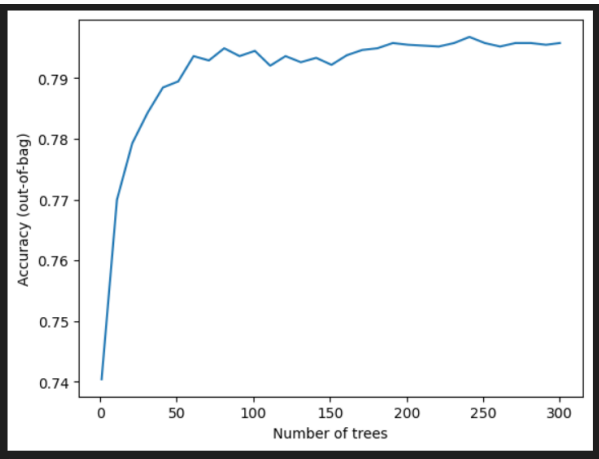
```
rf.fit(x=train_ds)
```

[21]

```
Reading training dataset...
Training dataset read in 0:00:04.789066. Found 6963 examples.
Training model...
Model trained in 0:00:50.250290
Compiling model...
Model compiled.

<tf_keras.src.callbacks.History at 0x798060425360>
```

```
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()
```




```
[24]
inspector = rf.make_inspector()
inspector.evaluation()
```

```
Evaluation(num_examples=6963, accuracy=0.7957776820336062, loss=0.5187134162947384, rmse=None, ndcg=None, aucs=None, auuc=None, qini=None)
```

```
[25]
evaluation = rf.evaluate(x=valid_ds,return_dict=True)

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

```
2/2 [==============================] - 1s 70ms/step - loss: 0.0000e+00 - accuracy: 0.7948
loss: 0.0000
accuracy: 0.7948
```

```
[26]
print(f"Available variable importances:")
for importance in inspector.variable_importances().keys():
  print("\t", importance)
```

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

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

```
[("CryoSleep" (1; #2), 92.0),
 ("Spa" (1; #10), 63.0),
 ("RoomService" (1; #7), 53.0),
 ("VRDeck" (1; #12), 44.0),
 ("ShoppingMall" (1; #8), 31.0),
 ("FoodCourt" (1; #5), 13.0),
 ("Deck" (4; #3), 2.0),
 ("HomePlanet" (4; #6), 2.0)]
```

```python
import pandas as pd
import tensorflow_decision_forests as tfdf

# Load the correct dataset
test_df = pd.read_csv('/kaggle/input/spaceship-titanic/test.csv')  # Use test.csv
submission_id = test_df.PassengerId

# Replace NaN values with zero
test_df[['VIP', 'CryoSleep']] = test_df[['VIP', 'CryoSleep']].fillna(value=0)

# Creating New Features - Deck, Cabin_num, and Side from the Cabin column
test_df[["Deck", "Cabin_num", "Side"]] = test_df["Cabin"].str.split("/", expand=True)
test_df = test_df.drop('Cabin', axis=1)

# Convert boolean to 1's and 0's
test_df['VIP'] = test_df['VIP'].astype(int)
test_df['CryoSleep'] = test_df['CryoSleep'].astype(int)

# Convert pd dataframe to tf dataset
test_ds = tfdf.keras.pd_dataframe_to_tf_dataset(test_df)

# Get the predictions for test data
predictions = rf.predict(test_ds)
n_predictions = (predictions > 0.5).astype(bool)

# Load sample submission
sample_submission_df = pd.read_csv('/kaggle/input/spaceship-titanic/sample_submission.csv')

# Ensure predictions match the sample submission length
n_predictions = n_predictions[:len(sample_submission_df)]

# Create output DataFrame
output = pd.DataFrame({'PassengerId': submission_id, 'Transported': n_predictions.squeeze()})
```

```python
# Create output DataFrame
output = pd.DataFrame({'PassengerId': submission_id, 'Transported': n_predictions.squeeze()})

# Save submission file
output.to_csv('/kaggle/working/submission.csv', index=False)

# Debugging
print(len(n_predictions))
print(len(sample_submission_df))
output.head()
```

[28]

```
/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 [==============================] - 0s 66ms/step
4277
4277
```

	PassengerId	Transported
0	0013_01	True
1	0018_01	False
2	0019_01	True
3	0021_01	True
4	0023_01	True

[29]
```
n_predictions = n_predictions[:len(sample_submission_df)]
```

[30]
```
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()
```

...

	PassengerId	Transported
0	0013_01	True
1	0018_01	False
2	0019_01	True
3	0021_01	True
4	0023_01	True