SUPERIOR UNIVERSITY

Submitted By:

Name: **Ali Raza**

Roll number **Su92-bsdsm-f23-018**

Section: **4A**

Task: **01**

Subject: **Programming for Ai (lab)**

Submitted to: **Sir Rasikh Ali**

https://www.kaggle.com/code/alijatt1/spaceship-titanic?scriptVersionId=224652938

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, key terms used, and its applications.

Overview of the Notebook

The notebook consists of the following key steps:

- **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, useful for handling arrays and mathematical operations.
2. **Pandas (pandas)** – A data manipulation library for reading, modifying, and analyzing tabular data.
3. **Seaborn (sns) and Matplotlib (plt)** – Libraries used for data visualization.
4. **TensorFlow (tensorflow)** – A machine learning library that supports deep learning models.
5. **TensorFlow Decision Forests (tensorflow_decision_forests)** – A library that implements decision forest models, which are used for classification and regression tasks.
6. **CSV (pd.read_csv)** – A common file format for storing tabular data. Pandas can load these files into a DataFrame for analysis.
7. **Exploratory Data Analysis (EDA)** – The process of understanding the dataset through summary statistics, visualizations, and data cleaning.
8. **Feature Engineering** – The process of creating new input features from existing data to improve model performance.
9. **Classification Task** – A machine learning problem where the goal is to categorize data (e.g., predicting whether a passenger was transported or not).
10. **Model Training** – The process of teaching a machine learning model to recognize patterns in the dataset.

11. **Evaluation Metrics** – Measures such as accuracy, precision, recall, and F1-score used to assess model performance.

Code Explanation

1. Importing Libraries

The notebook begins by importing essential Python 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**

These libraries are used for **data processing, visualization, and machine learning**.

2. Installed Versions

To ensure compatibility, the notebook prints the versions of TensorFlow and TFDF:

print("TensorFlow v" + tf.__version__)

print("TensorFlow Decision Forests v" + tfdf.__version__)

3. Loading the Dataset

The dataset is loaded from Kaggle into a Pandas DataFrame:

dataset_df = pd.read_csv('/kaggle/input/spaceship-titanic/train.csv')

print("Full train dataset shape is {}".format(dataset_df.shape))

This confirms the dataset has been successfully loaded.

4. Data Exploration and Preprocessing

- **Summary Statistics:**
- dataset_df.describe()

Provides an overview of numeric columns, including mean, standard deviation, and range.

- **Dataset Information:**
- dataset_df.info()

Displays column names, data types, and missing values.

- **Handling Missing Values:**
  The notebook likely includes strategies for handling missing data, such as **imputation** (filling missing values with mean/median/mode) or **dropping missing rows/columns**.
- **Encoding Categorical Variables:**
  Machine learning models require numerical data. The notebook converts categorical features into numbers using encoding techniques.
- **Feature Scaling:**
  Some numerical features may be scaled to improve model performance.

5. Feature Engineering

The notebook may extract meaningful information from existing features, such as:

- Creating new features based on **domain knowledge**.
- Transforming categorical variables into numerical values.
- Combining multiple features to generate better inputs for the model.

6. Model Selection and Training

- **Using TensorFlow Decision Forests for Classification**
- model = tfdf.keras.RandomForestModel()

Decision forests use multiple decision trees to make predictions.

- **Training the Model**
- model.fit(train_data, train_labels)

The model is trained using the processed dataset.

- **Hyperparameter Tuning**
  The notebook may adjust model parameters to improve accuracy.

7. Model Evaluation

The model's performance is measured using classification metrics:

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

y_pred = model.predict(test_data)

accuracy = accuracy_score(test_labels, y_pred)

print(f"Model Accuracy: {accuracy}")

Other possible evaluation techniques include:

- **Confusion Matrix** – Visualizing true positives, false positives, etc.
- **Precision, Recall, and F1-Score** – Assessing model reliability.

8. Prediction Process

- **Making Predictions on New Data**
- predictions = model.predict(test_data)

The trained model predicts whether a passenger was transported.

- **Saving Predictions for Submission**
- submission = pd.DataFrame({'PassengerId': test_data['PassengerId'], 'Transported': predictions})
- submission.to_csv('submission.csv', index=False)

The predictions are stored in a CSV file for Kaggle submission.

Applications of the Notebook

This notebook demonstrates **machine learning applications in classification tasks**. Key applications include:

- **Predictive Analytics:** Identifying trends and making forecasts.
- **Data Cleaning & Preprocessing:** Handling missing values, encoding categorical data.
- **Model Deployment:** Applying trained models to real-world problems.
- **Feature Engineering:** Extracting meaningful insights from raw data.
- **Deep Learning Implementation:** Using advanced techniques to enhance accuracy.

Conclusion

The **Spaceship Titanic** notebook provides a structured approach to solving a classification problem using **TensorFlow Decision Forests**. The key steps include:

- **Loading and preprocessing data**
- **Exploratory Data Analysis (EDA)**
- **Feature engineering**
- **Training and evaluating a machine learning model**
- **Making predictions and saving results**

```python
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
```

```python
print("TensorFlow v" + tf.__version__)
print("TensorFlow Decision Forests v" + tfdf.__version__)
```

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

```python
# Load a dataset into a Pandas Dataframe
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)
```

```python
dataset_df.describe()
```

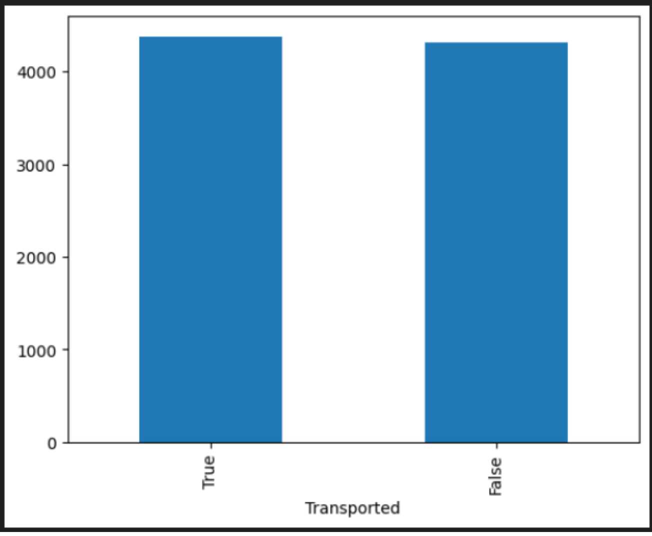
	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	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

```python
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
 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")
```

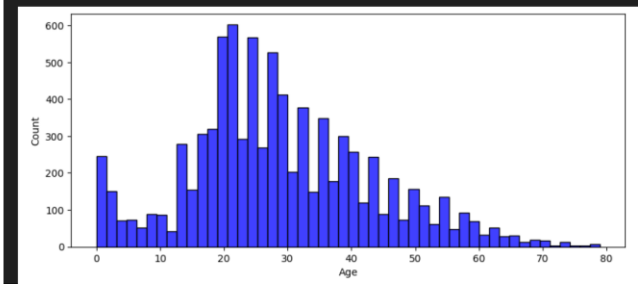
```
<Axes: xlabel='Transported'>
```

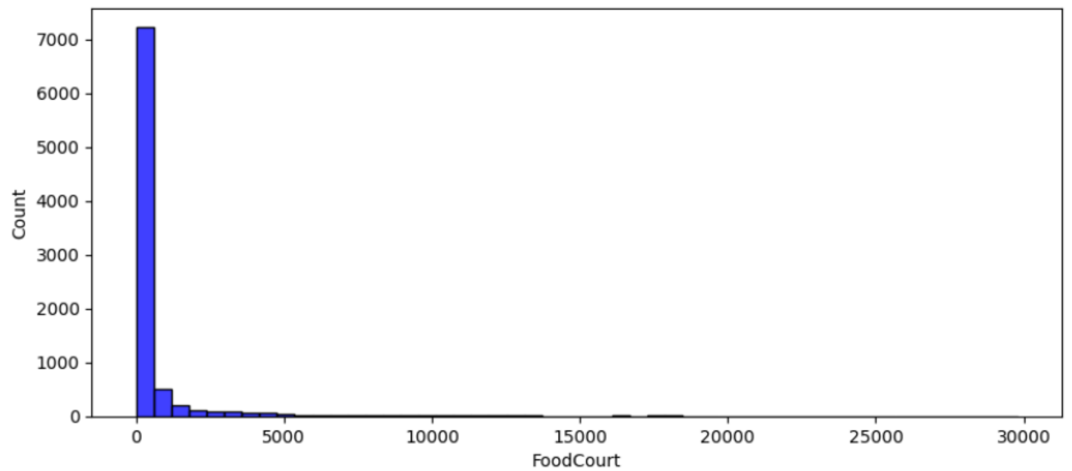


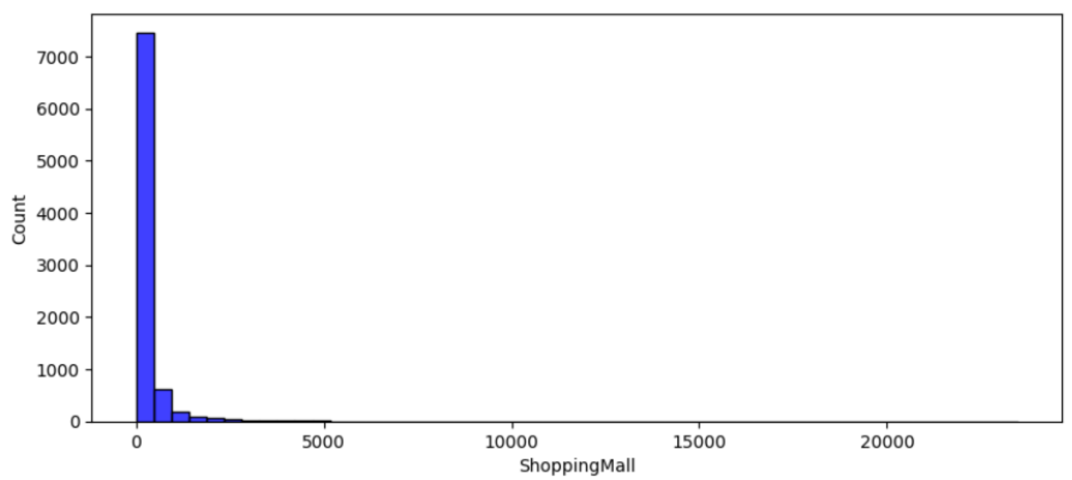
```python
fig, ax = plt.subplots(5,1,  figsize=(10, 10))
plt.subplots_adjust(top = 2)

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]);
```

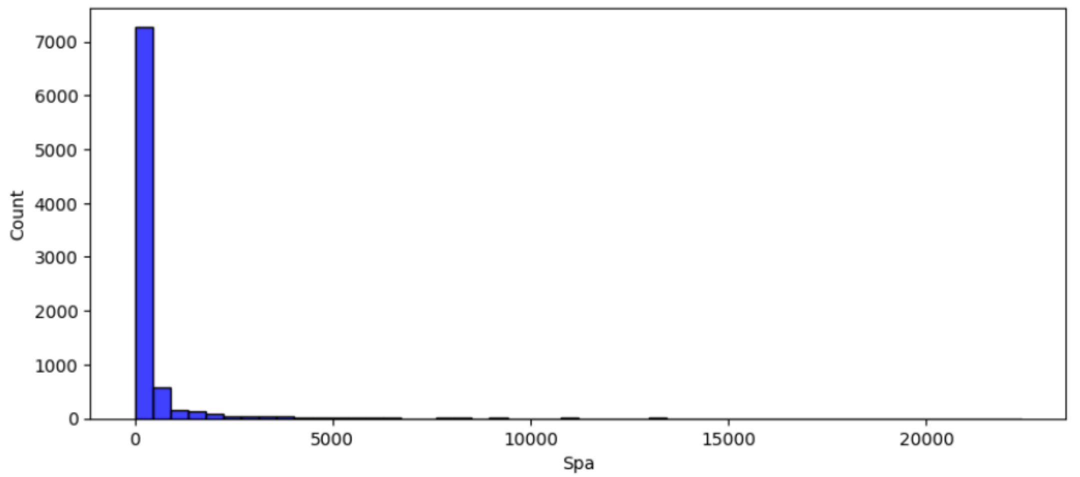
Python

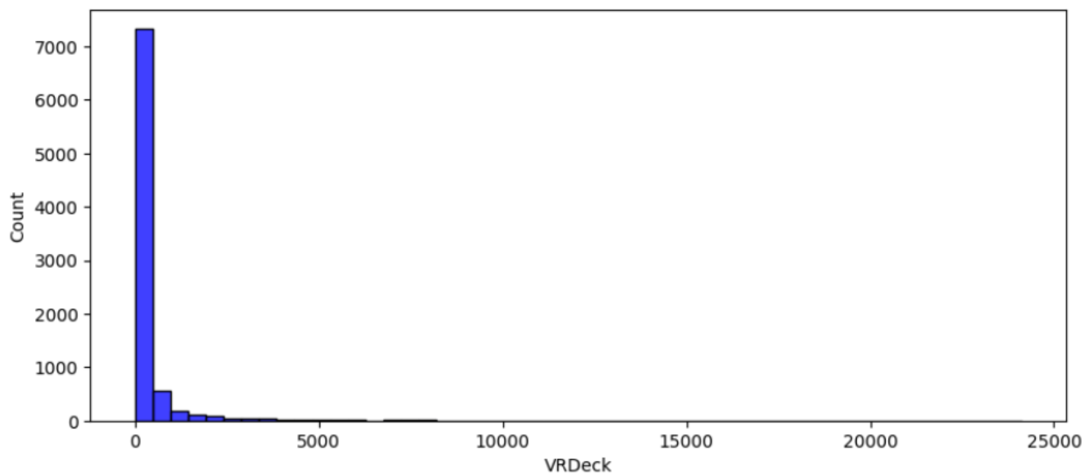
```
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: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











[8]
```python
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

[9]
```python
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
```

```python
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)
```

```
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
```

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

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

```python
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")
```

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

	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

```python
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)))
```

```
6977 examples in training, 1716 examples in testing.
```

```python
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)
```

```python
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]
```

```
rf = tfdf.keras.RandomForestModel()
rf.compile(metrics=["accuracy"]) # Optional, you can use this to include a list of eval metrics
```

```
Use /tmp/tmp6_6y4q_z as temporary training directory
```

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

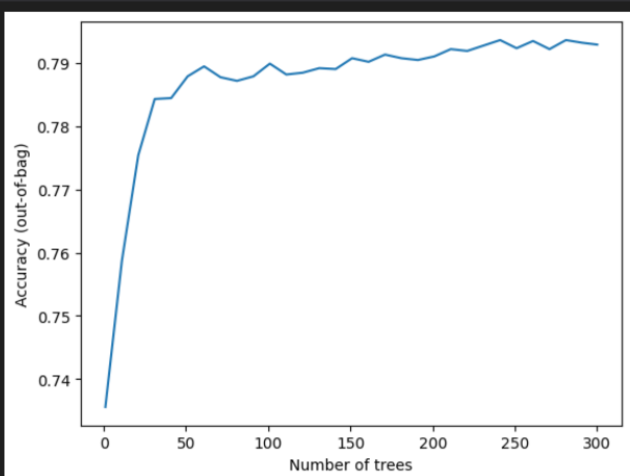
```
Reading training dataset...
Training dataset read in 0:00:03.509253. Found 6977 examples.
Training model...
Model trained in 0:00:28.672398
Compiling model...
Model compiled.

<tf_keras.src.callbacks.History at 0x78235aebf4f0>
```

```
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()
```

[22]

...




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

[23]

```
Evaluation(num_examples=6977, accuracy=0.7928909273326644, loss=0.5584456459319431, 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}")
```

24]

```
2/2 [==============================] - 1s 59ms/step - loss: 0.0000e+00 - accuracy: 0.8170
loss: 0.0000
accuracy: 0.8170
```

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

25]

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

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

26]

```
[("CryoSleep" (1; #2), 128.0),
 ("Spa" (1; #10), 54.0),
 ("RoomService" (1; #7), 52.0),
 ("VRDeck" (1; #12), 36.0),
 ("ShoppingMall" (1; #8), 21.0),
 ("FoodCourt" (1; #5), 6.0),
 ("Deck" (4; #3), 3.0)]
```

```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()
```

[27]

```
1/5 [=====>........................] - ETA: 0s
/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 60ms/step
```

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

```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()
```

[28]

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