Wrocław University of Science and Technology

PYTHON
LABORATORY REPORT

Faculty of Electronics, Photonics and MicrosystemsPYTHON LABORATORY

Theme of class: Data Analysis and Machine Learning
Student: Hayrettin Aycetin (276807)
Date of class: 18.12.2023 15:15-16:55
Group No:3
Submission Date:07.01.2024

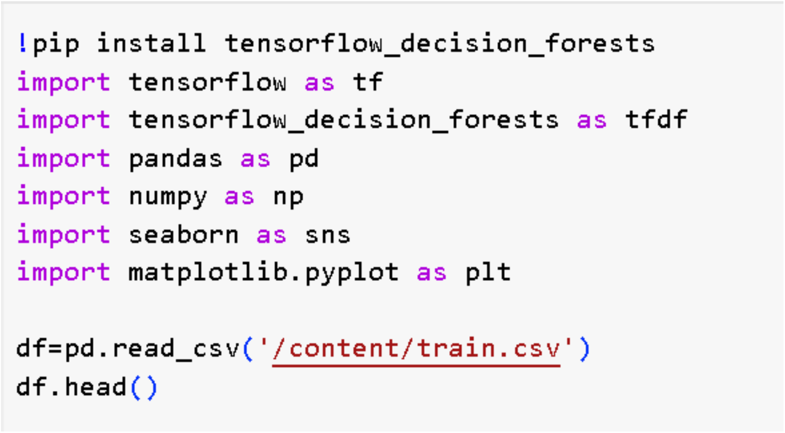
Lab assistant: Aleksander Kubeczek, Alicja Kwaśny GRADE:

Task 1

Load the dataset:

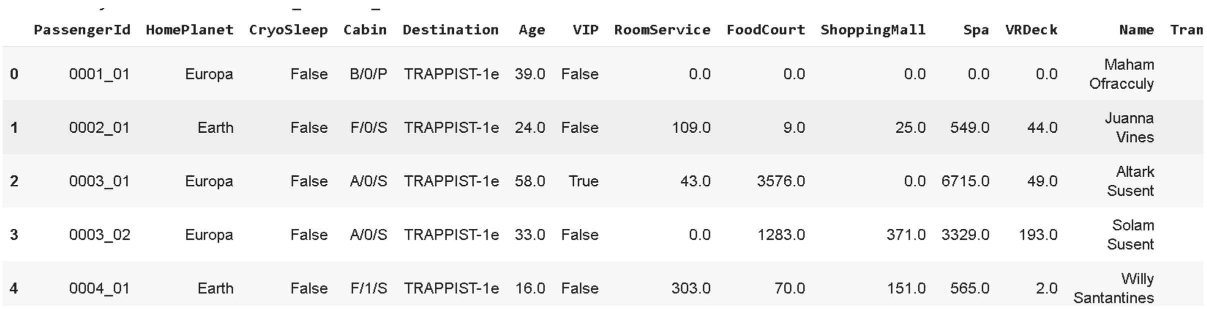
- Read the csv file (you can use the read_csv function from pandas)
- Display 5 first entries of the dataset to check whether everything loaded correctly

Code:



```
!pip install tensorflow_decision_forests
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

df=pd.read_csv('/content/train.csv')
df.head()
```

Output:

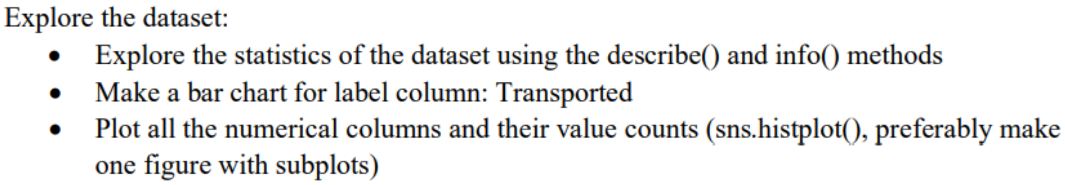


	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Tran
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	Maham Ofracculy	
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	Juanna Vines	
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	Altark Susent	
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	
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	

Comments:

Firstly I imported all needed library for this list .Using read_csv() function I read the dataset and assigned as df.I used head() function to read the first five entries of the dataframe. This part was easy and straightforward.

Task 2

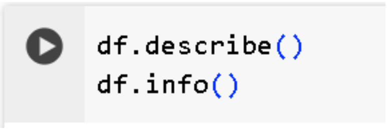


Explore the dataset:

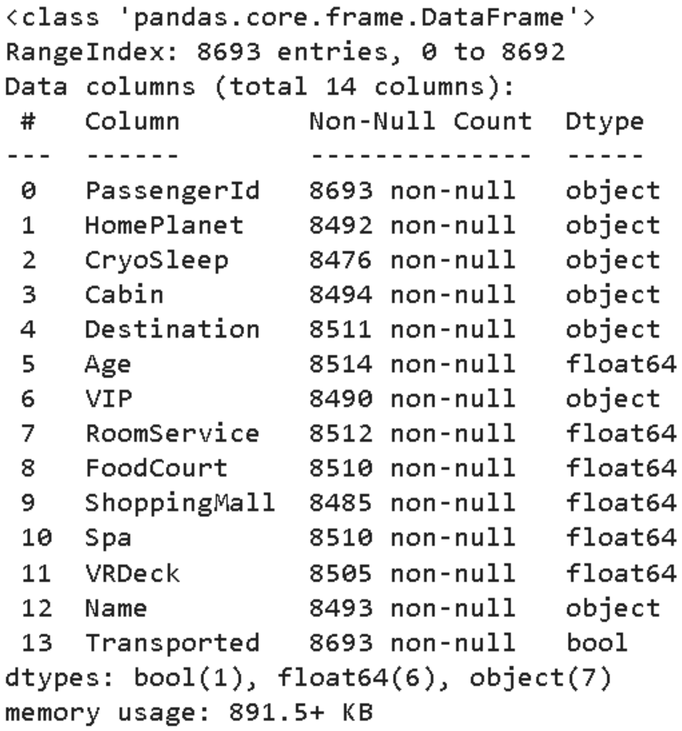
- Explore the statistics of the dataset using the describe() and info() methods
- Make a bar chart for label column: Transported
- Plot all the numerical columns and their value counts (sns.histplot(), preferably make one figure with subplots)

Codes:

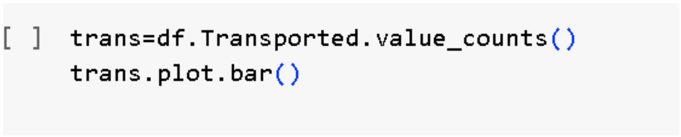
Outputs:



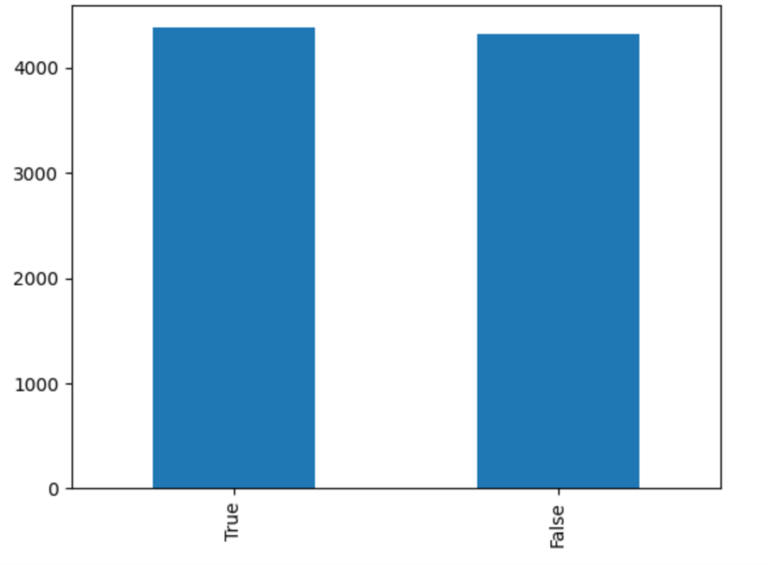
```
df.describe()
df.info()
```

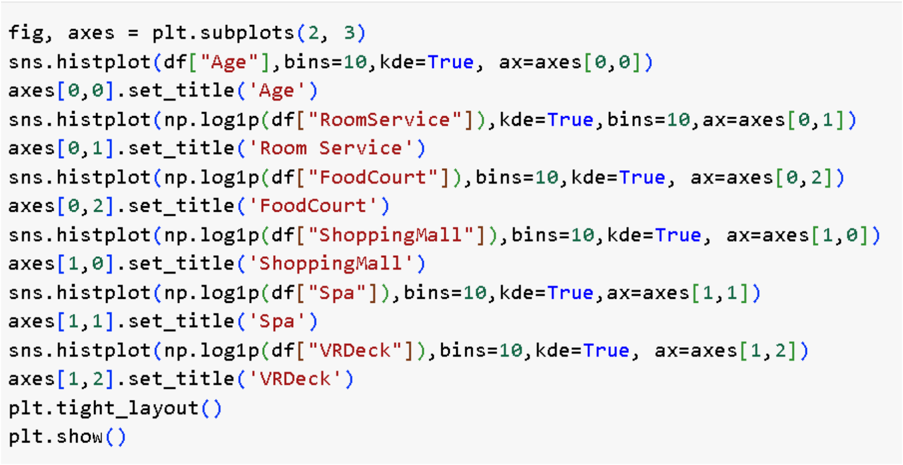


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   PassengerId   8693 non-null   object 
 1   HomePlanet    8492 non-null   object 
 2   CryoSleep   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

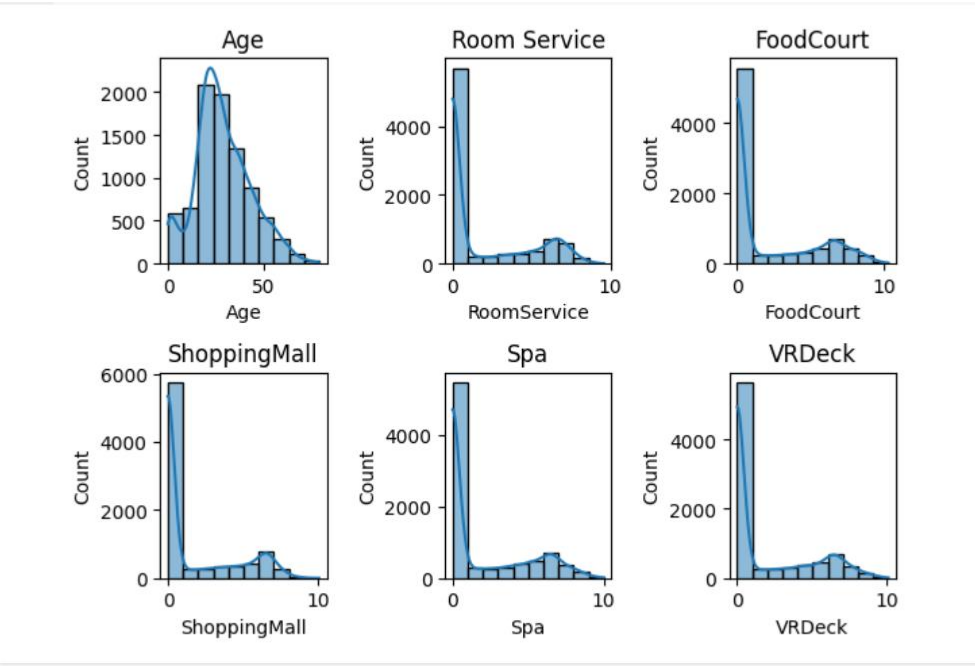


```
[ ]  trans=df.Transported.value_counts()
     trans.plot.bar()
```





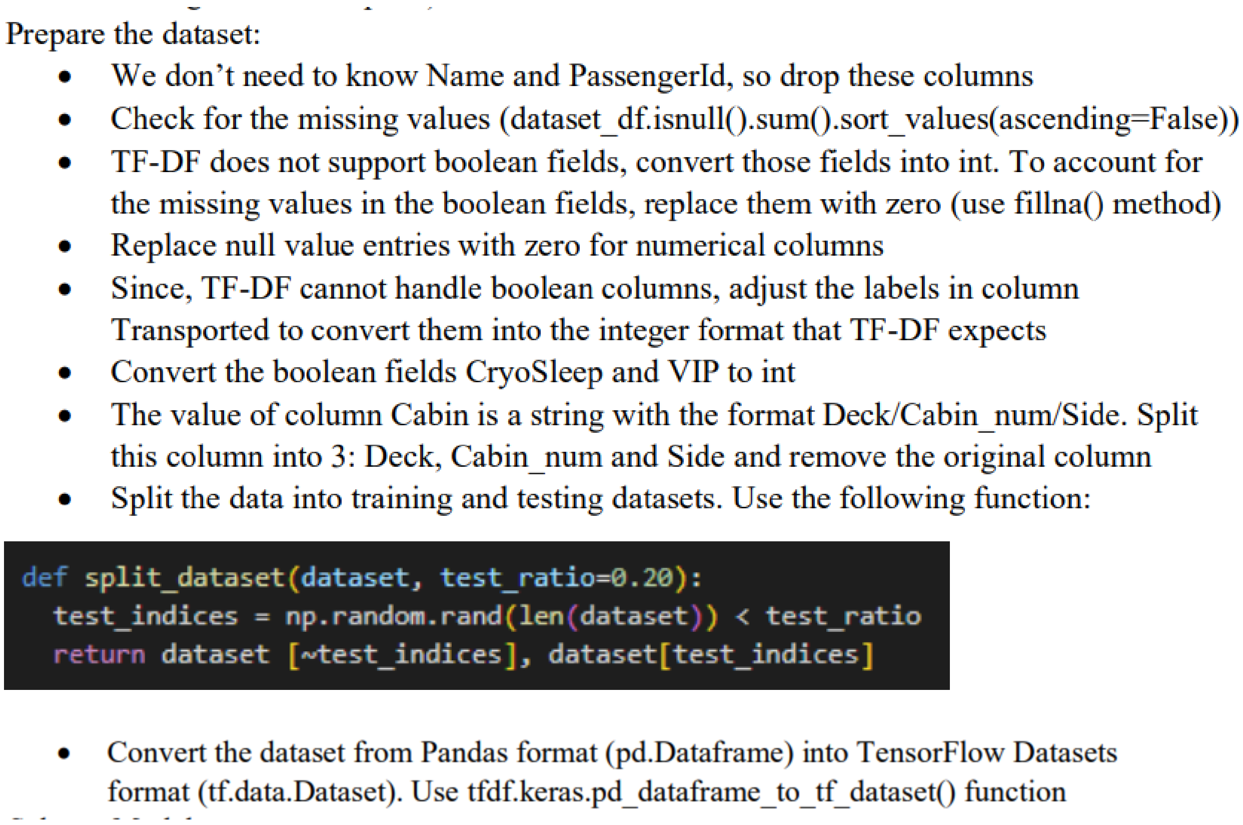
```
fig, axes = plt.subplots(2, 3)
sns.histplot(df["Age"],bins=10,kde=True, ax=axes[0,0])
axes[0,0].set_title('Age')
sns.histplot(np.log1p(df["RoomService"]),kde=True,bins=10,ax=axes[0,1])
axes[0,1].set_title('Room Service')
sns.histplot(np.log1p(df["FoodCourt"]),bins=10,kde=True, ax=axes[0,2])
axes[0,2].set_title('FoodCourt')
sns.histplot(np.log1p(df["ShoppingMall"]),bins=10,kde=True, ax=axes[1,0])
axes[1,0].set_title('ShoppingMall')
sns.histplot(np.log1p(df["Spa"]),bins=10,kde=True,ax=axes[1,1])
axes[1,1].set_title('Spa')
sns.histplot(np.log1p(df["VRDeck"]),bins=10,kde=True, ax=axes[1,2])
axes[1,2].set_title('VRDeck')
plt.tight_layout()
plt.show()
```



Comments:

Firstly I used describe() and info() functions to explore the dataset.
Later I made a bar graph from transported people’s value counts which is boolean.
For the last part I used subplots() function in pandas and set my subplots like an 2 rows 3 columns array.By using histplot() function and its parameters such as data,bins,kde and ax from seaborn library I made my plots.Bins controls the granularity of the histogram.Kde provides a smooth estimate of the distribution as a line on it and ax determines where the plots will be on the subplot array. The most important part was np.log1p(column): Applied the natural logarithm plus one (log(1 + x)) to the values in the column. This transformation is used for dealing with data that includes zero values, as it handles them more gracefully.

Task 3



Prepare the dataset:

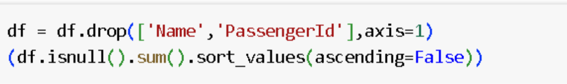
- We don’t need to know Name and PassengerId, so drop these columns
- Check for the missing values (dataset_df.isnull().sum().sort_values(ascending=False))
- TF-DF does not support boolean fields, convert those fields into int. To account for the missing values in the boolean fields, replace them with zero (use fillna() method)
- Replace null value entries with zero for numerical columns
- Since, TF-DF cannot handle boolean columns, adjust the labels in column Transported to convert them into the integer format that TF-DF expects
- Convert the boolean fields CryoSleep and VIP to int
- The value of column Cabin is a string with the format Deck/Cabin_num/Side. Split this column into 3: Deck, Cabin_num and Side and remove the original column
- Split the data into training and testing datasets. Use the following function:

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

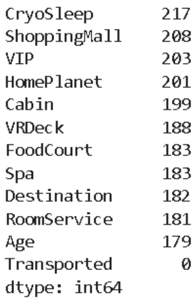
- Convert the dataset from Pandas format (pd.Dataframe) into TensorFlow Datasets format (tf.data.Dataset). Use tfdf.keras.pd_dataframe_to_tf_dataset() function

Codes:

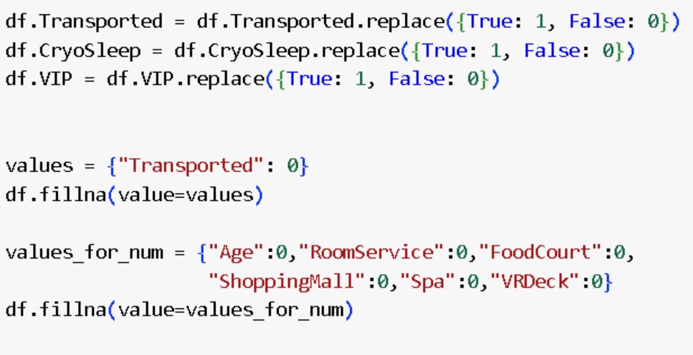
Outputs:



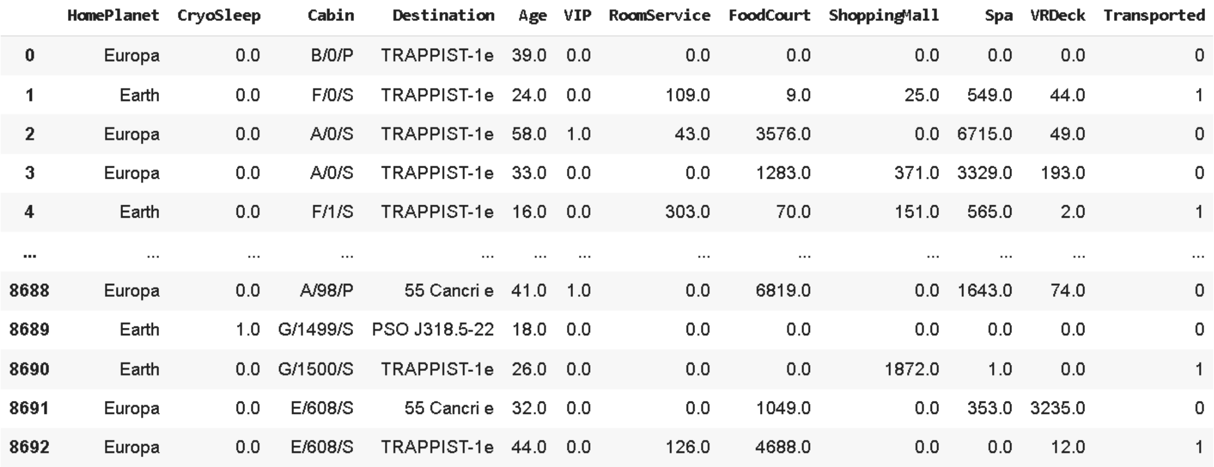
```
df = df.drop(['Name','PassengerId'],axis=1)
(df.isnull().sum().sort_values(ascending=False))
```



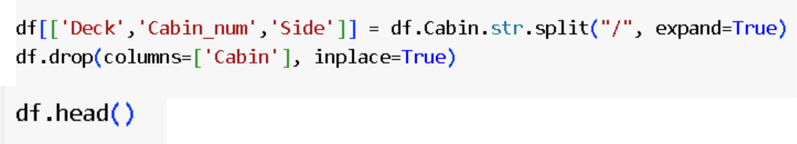
```
CryoSleep       217
ShoppingMall    208
VIP             203
HomePlanet      201
Cabin           199
VRDeck          188
FoodCourt       183
Spa             183
Destination     182
RoomService     181
Age             179
Transported       0
dtype: int64
```



```
df.Transported = df.Transported.replace({True: 1, False: 0})
df.CryoSleep = df.CryoSleep.replace({True: 1, False: 0})
df.VIP = df.VIP.replace({True: 1, False: 0})


values = {"Transported": 0}
df.fillna(value=values)

values_for_num = {"Age":0,"RoomService":0,"FoodCourt":0,
                  "ShoppingMall":0,"Spa":0,"VRDeck":0}
df.fillna(value=values_for_num)
```

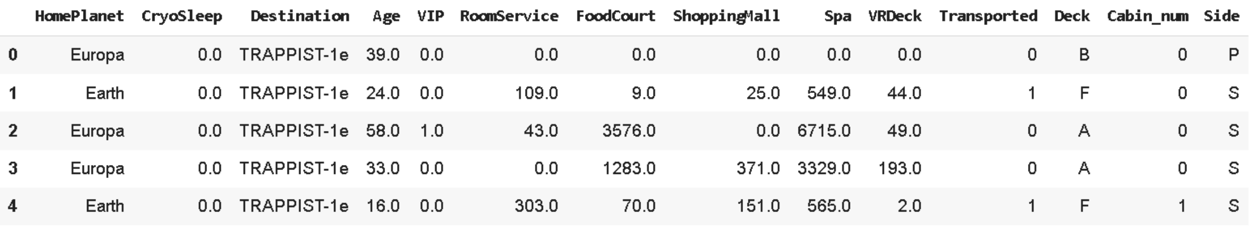


	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Transported
0	Europa	0.0	B/0/P	TRAPPIST-1e	39.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	Earth	0.0	F/0/S	TRAPPIST-1e	24.0	0.0	109.0	9.0	25.0	549.0	44.0	1
2	Europa	0.0	A/0/S	TRAPPIST-1e	58.0	1.0	43.0	3576.0	0.0	6715.0	49.0	0
3	Europa	0.0	A/0/S	TRAPPIST-1e	33.0	0.0	0.0	1283.0	371.0	3329.0	193.0	0
4	Earth	0.0	F/1/S	TRAPPIST-1e	16.0	0.0	303.0	70.0	151.0	565.0	2.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
8688	Europa	0.0	A/98/P	55 Cancri e	41.0	1.0	0.0	6819.0	0.0	1643.0	74.0	0
8689	Earth	1.0	G/1499/S	PSO J318.5-22	18.0	0.0	0.0	0.0	0.0	0.0	0.0	0
8690	Earth	0.0	G/1500/S	TRAPPIST-1e	26.0	0.0	0.0	0.0	1872.0	1.0	0.0	1
8691	Europa	0.0	E/608/S	55 Cancri e	32.0	0.0	0.0	1049.0	0.0	353.0	3235.0	0
8692	Europa	0.0	E/608/S	TRAPPIST-1e	44.0	0.0	126.0	4688.0	0.0	0.0	12.0	1

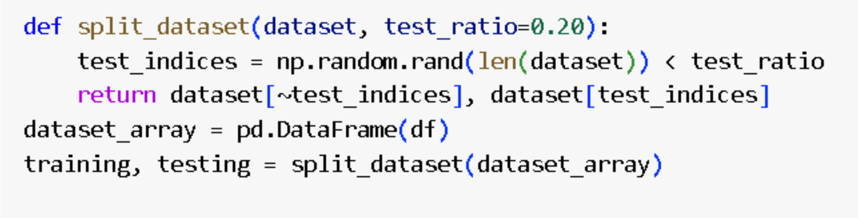


```
df[['Deck','Cabin_num','Side']] = df.Cabin.str.split("/", expand=True)
df.drop(columns=['Cabin'], inplace=True)
```

```
df.head()
```

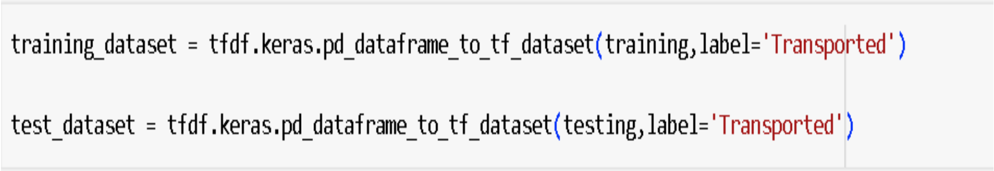


	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Transported	Deck	Cabin_num	Side
0	Europa	0.0	TRAPPIST-1e	39.0	0.0	0.0	0.0	0.0	0.0	0.0	0	B	0	P
1	Earth	0.0	TRAPPIST-1e	24.0	0.0	109.0	9.0	25.0	549.0	44.0	1	F	0	S
2	Europa	0.0	TRAPPIST-1e	58.0	1.0	43.0	3576.0	0.0	6715.0	49.0	0	A	0	S
3	Europa	0.0	TRAPPIST-1e	33.0	0.0	0.0	1283.0	371.0	3329.0	193.0	0	A	0	S
4	Earth	0.0	TRAPPIST-1e	16.0	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]
dataset_array = pd.DataFrame(df)
training, testing = split_dataset(dataset_array)
```

No output



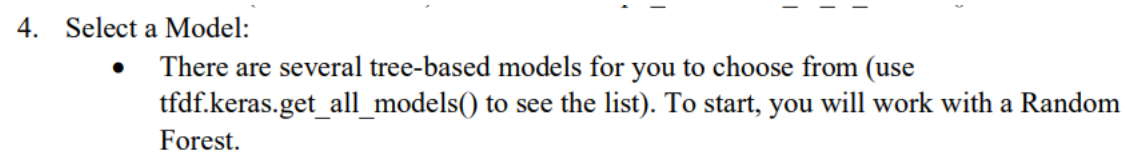
```
training_dataset = tfdf.keras.pd_dataframe_to_tf_dataset(training,label='Transported')

test_dataset = tfdf.keras.pd_dataframe_to_tf_dataset(testing,label='Transported')
```

No output

Comments:

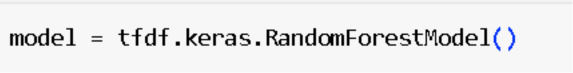
This part was mainly about preparing and cleaning the dataset for tensorflow.Firstly I dropped name and passenger_id colmns using drop() function and later checked missing values using given function.Later I replaced boolean values with their integer values such as True:1 and False:0 using replace() function.Then using fillna() function I filled empty values with zero.Later I splitted Cabin column using split() function to wanted columns and then dropped it.Lastly I split dataset to training and testing with given split_dataset() function and using tfdf.keras.pd_dataframe_to_tf_dataset() function I converted pandas dataframes of testing and training to tensorflow dataframe and I used label = “Transported ” because that’s where we need to predict.

Task 4



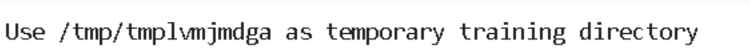
4. Select a Model:
   - There are several tree-based models for you to choose from (use tfdf.keras.get_all_models() to see the list). To start, you will work with a Random Forest.

Code:



```
model = tfdf.keras.RandomForestModel()
```

Output:

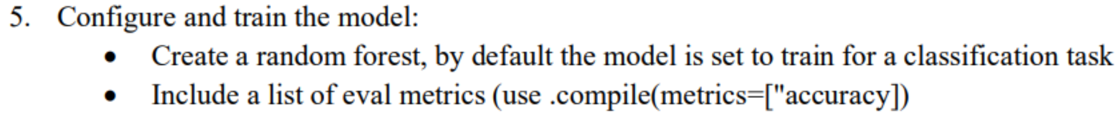


```
Use /tmp/tmplvmjmdga as temporary training directory
```

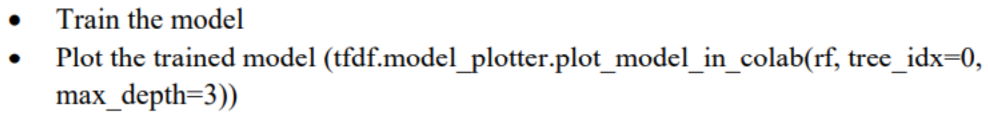
Comments:

In this part I just choose a model which is RandomForestModel as wanted in list and I assigned it to model.

Task 5

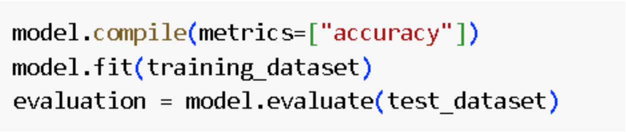


5. Configure and train the model:
   - Create a random forest, by default the model is set to train for a classification task
   - Include a list of eval metrics (use .compile(metrics=["accuracy])



- Train the model
- Plot the trained model (tfdf.model_plotter.plot_model_in_colab(rf, tree_idx=0, max_depth=3))

Code:

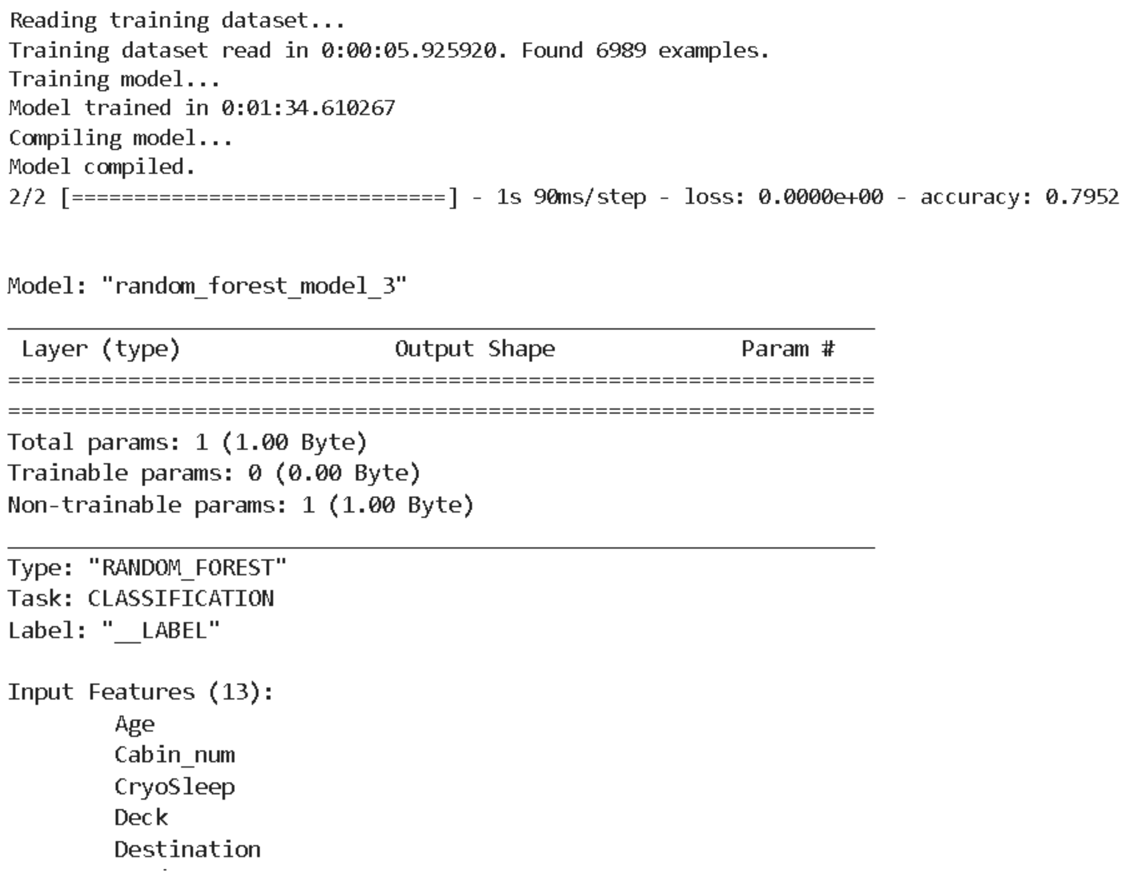


```
model.compile(metrics=["accuracy"])
model.fit(training_dataset)
evaluation = model.evaluate(test_dataset)
```

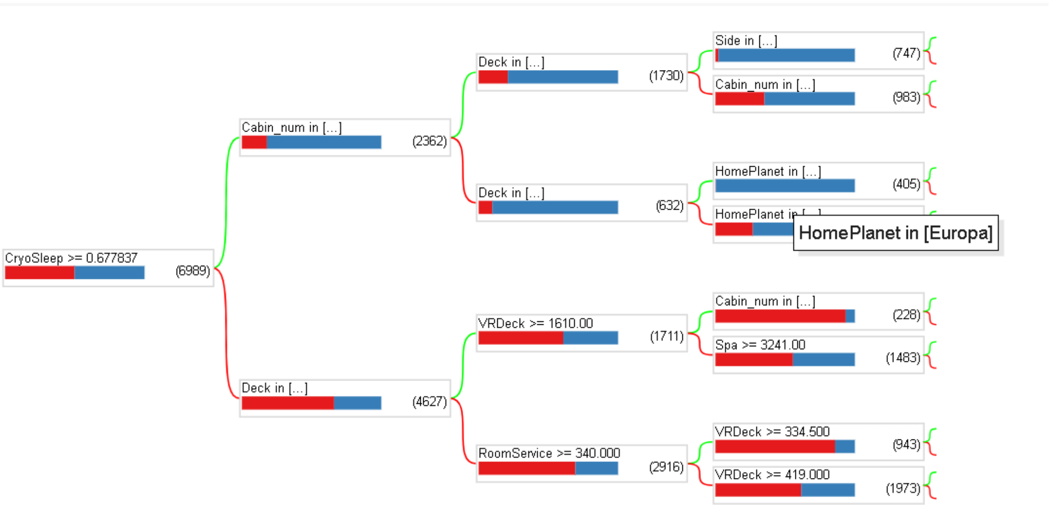
```
model.summary()
```

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

Outputs:



```
Reading training dataset...
Training dataset read in 0:00:05.925920. Found 6989 examples.
Training model...
Model trained in 0:01:34.610267
Compiling model...
Model compiled.
2/2 [==============================] - 1s 90ms/step - loss: 0.0000e+00 - accuracy: 0.7952


Model: "random_forest_model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
=================================================================
Total params: 1 (1.00 Byte)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 1 (1.00 Byte)
_________________________________________________________________
Type: "RANDOM_FOREST"
Task: CLASSIFICATION
Label: "__LABEL"

Input Features (13):
	Age
	Cabin_num
	CryoSleep
	Deck
	Destination
```

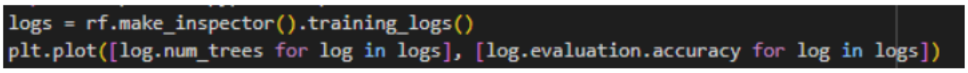


Comments:

Firstly I created a random forest model in task 4 so I just continued here with the including a list of eval metrics given in the task and I used model.fit() to train the model.Later I used model.evaluate() to return a tuple of evaluation results.Lastly I used the given plotting to plot my model.

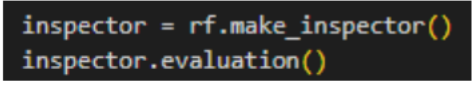
Task 6

Evaluate the model on the Out of bag (OOB) data and validation dataset



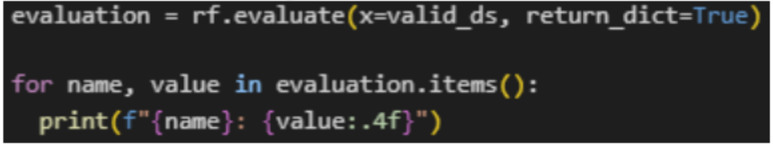
```
logs = rf.make_inspector().training_logs()
plt.plot([log.num_trees for log in logs], [log.evaluation.accuracy for log in logs])
```

- Plot the accuracy evaluated on the out-of-bag dataset according to the number of trees in the model.
- See some general statistics on the OOB dataset:



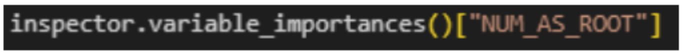
```
inspector = rf.make_inspector()
inspector.evaluation()
```

- Run the evaluation using the validation dataset



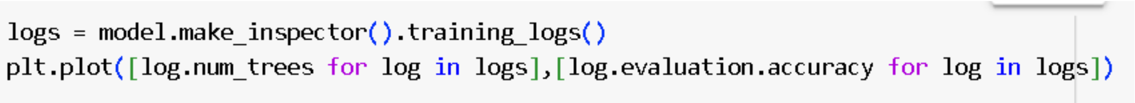
```
evaluation = rf.evaluate(x=valid_ds, return_dict=True)

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

- Display the important features:

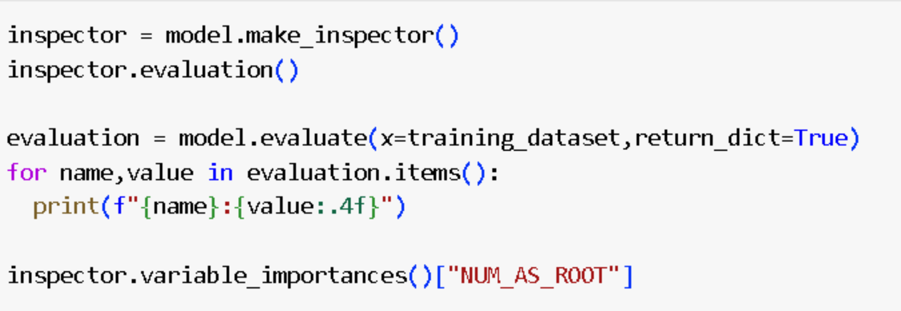


```
inspector.variable_importances()["NUM_AS_ROOT"]
```

Code:



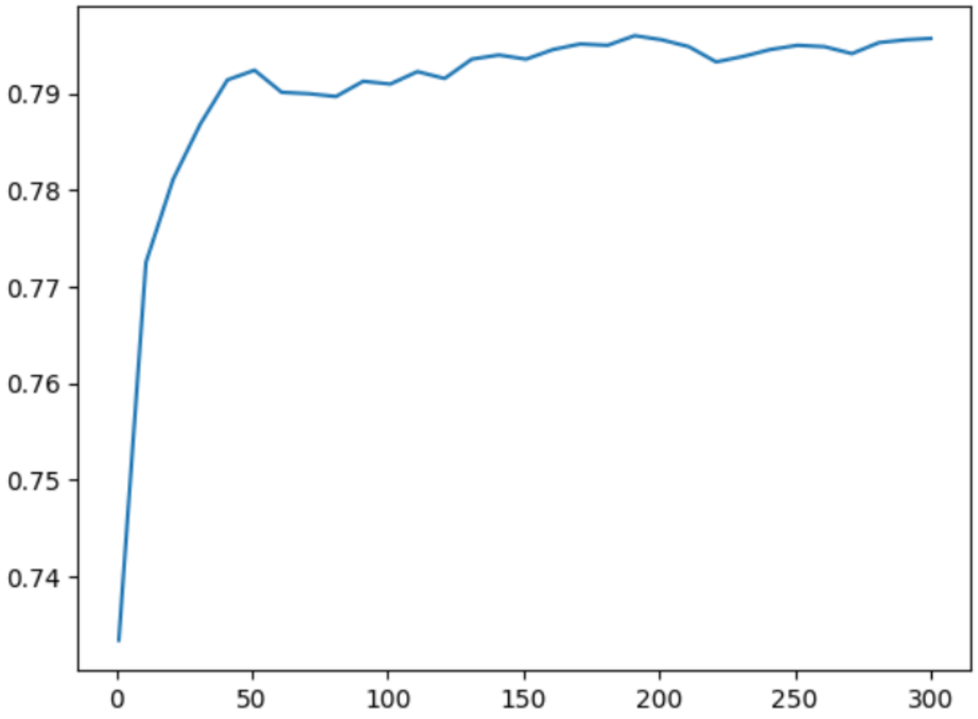
```
logs = model.make_inspector().training_logs()
plt.plot([log.num_trees for log in logs],[log.evaluation.accuracy for log in logs])
```

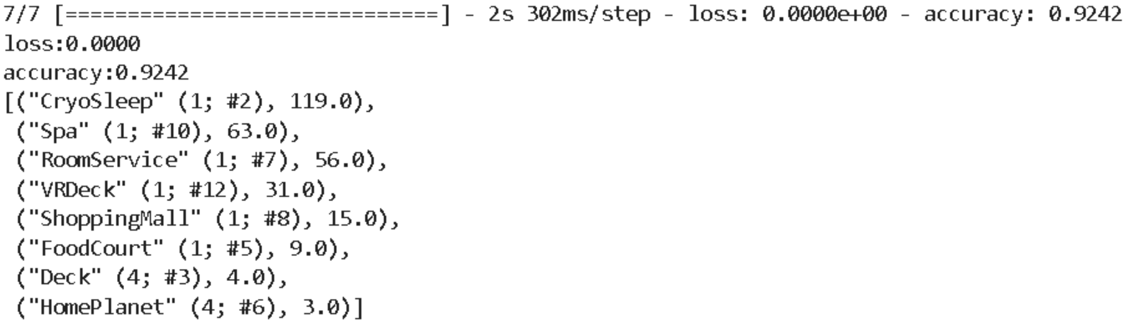


```
inspector = model.make_inspector()
inspector.evaluation()

evaluation = model.evaluate(x=training_dataset,return_dict=True)
for name,value in evaluation.items():
  print(f"{name}:{value:.4f}")

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

Outputs:

```
[<matplotlib.lines.Line2D at 0x790117e48790>]
```





```
7/7 [==============================] - 2s 302ms/step - loss: 0.0000e+00 - accuracy: 0.9242
loss:0.0000
accuracy:0.9242
[("CryoSleep" (1; #2), 119.0),
 ("Spa" (1; #10), 63.0),
 ("RoomService" (1; #7), 56.0),
 ("VRDeck" (1; #12), 31.0),
 ("ShoppingMall" (1; #8), 15.0),
 ("FoodCourt" (1; #5), 9.0),
 ("Deck" (4; #3), 4.0),
 ("HomePlanet" (4; #6), 3.0)]
```

Comments:

In this part mostly the whole code was given by instructor in the list I just changed rf to model.This part was a bit hard to understand but as long as I understood the first code snippet inspects and evaluates a machine learning model by using an inspector to check the model's evaluation and variable importances. The evaluation results, including various metrics, are printed, and the importance of a specific variable ("NUM_AS_ROOT") is retrieved.

Link of my Collab:

https://colab.research.google.com/drive/13CCqkFQ5MkC5FO6O5WzAjdC3kwja8sLz?usp=sharing

Conclusion:

In general it was an really interesting and fun lab.Also it was really helpful about data analysis and machine learning basics.