```
import tensorflow as tf
import tensorflow_decision_forests as tfdf
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

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

```
Full train dataset shape is (1460, 81)
```

```
dataset = dataset.drop('Id', axis=1)
dataset.head(3)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     1460 non-null   int64
 1   MSZoning       1460 non-null   object
 2   LotFrontage    1201 non-null   float64
 3   LotArea        1460 non-null   int64
 4   Street         1460 non-null   object
 5   Alley          91 non-null     object
 6   LotShape       1460 non-null   object
 7   LandContour    1460 non-null   object
 8   Utilities      1460 non-null   object
 9   LotConfig      1460 non-null   object
 10  LandSlope      1460 non-null   object
 11  Neighborhood   1460 non-null   object
 12  Condition1     1460 non-null   object
 13  Condition2     1460 non-null   object
 14  BldgType       1460 non-null   object
 15  HouseStyle     1460 non-null   object
 16  OverallQual    1460 non-null   int64
 17  OverallCond    1460 non-null   int64
 18  YearBuilt      1460 non-null   int64
 19  YearRemodAdd   1460 non-null   int64
 20  RoofStyle      1460 non-null   object
 21  RoofMatl       1460 non-null   object
 22  Exterior1st    1460 non-null   object
 23  Exterior2nd    1460 non-null   object
 24  MasVnrType     588 non-null    object
 25  MasVnrArea     1452 non-null   float64
 26  ExterQual      1460 non-null   object
 27  ExterCond      1460 non-null   object
 28  Foundation     1460 non-null   object
 29  BsmtQual       1423 non-null   object
```

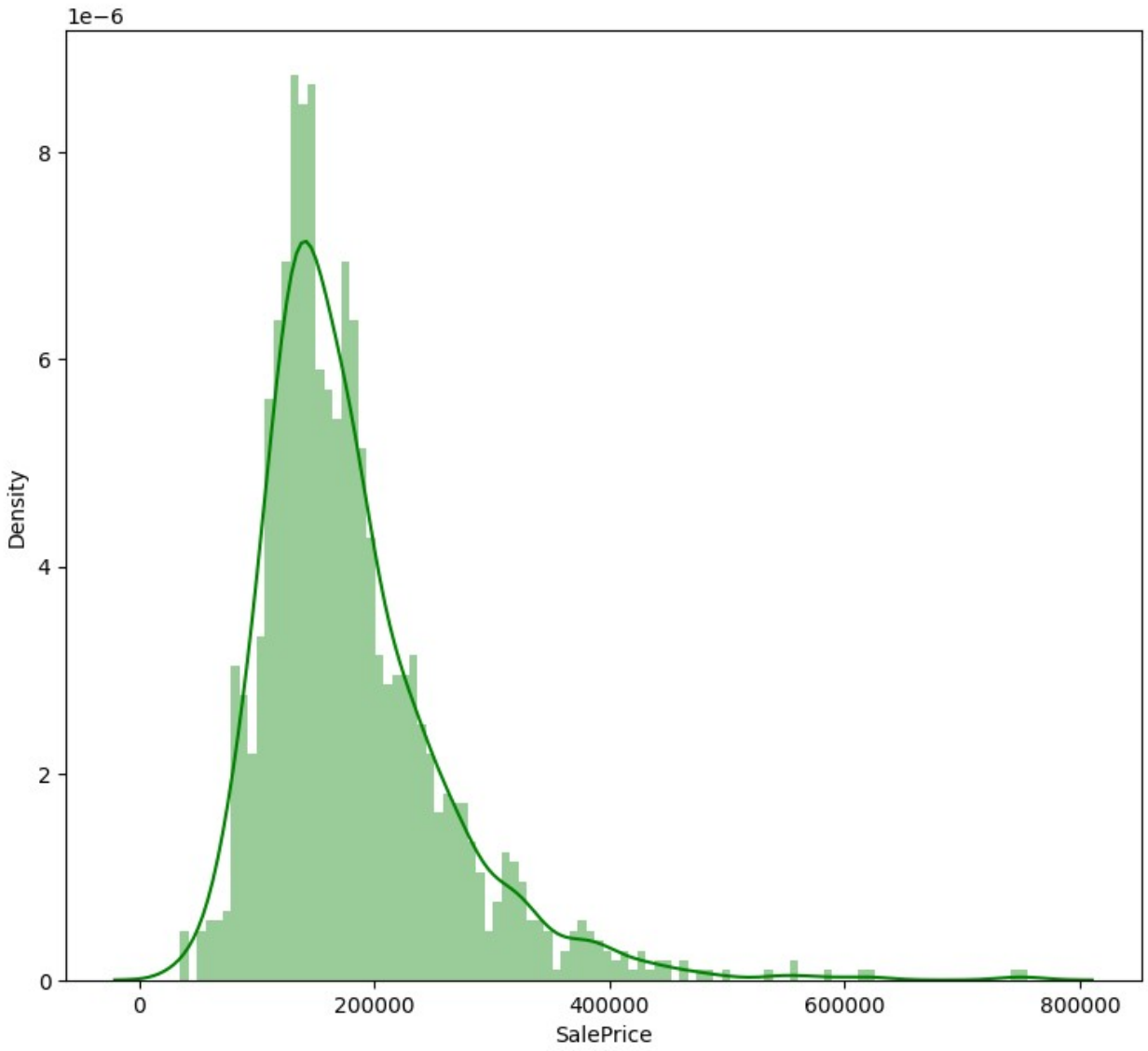
```
30  BsmtCond       1423 non-null   object
31  BsmtExposure   1422 non-null   object
32  BsmtFinType1   1423 non-null   object
33  BsmtFinSF1     1460 non-null   int64
34  BsmtFinType2   1422 non-null   object
35  BsmtFinSF2     1460 non-null   int64
36  BsmtUnfSF      1460 non-null   int64
37  TotalBsmtSF    1460 non-null   int64
38  Heating        1460 non-null   object
39  HeatingQC      1460 non-null   object
40  CentralAir     1460 non-null   object
41  Electrical     1459 non-null   object
42  1stFlrSF       1460 non-null   int64
43  2ndFlrSF       1460 non-null   int64
44  LowQualFinSF   1460 non-null   int64
45  GrLivArea      1460 non-null   int64
46  BsmtFullBath   1460 non-null   int64
47  BsmtHalfBath   1460 non-null   int64
48  FullBath       1460 non-null   int64
49  HalfBath       1460 non-null   int64
50  BedroomAbvGr   1460 non-null   int64
51  KitchenAbvGr   1460 non-null   int64
52  KitchenQual    1460 non-null   object
53  TotRmsAbvGrd   1460 non-null   int64
54  Functional     1460 non-null   object
55  Fireplaces     1460 non-null   int64
56  FireplaceQu    770 non-null    object
57  GarageType     1379 non-null   object
58  GarageYrBlt    1379 non-null   float64
59  GarageFinish   1379 non-null   object
60  GarageCars     1460 non-null   int64
61  GarageArea     1460 non-null   int64
62  GarageQual     1379 non-null   object
63  GarageCond     1379 non-null   object
64  PavedDrive     1460 non-null   object
65  WoodDeckSF     1460 non-null   int64
66  OpenPorchSF    1460 non-null   int64
67  EnclosedPorch  1460 non-null   int64
68  3SsnPorch      1460 non-null   int64
69  ScreenPorch    1460 non-null   int64
70  PoolArea       1460 non-null   int64
71  PoolQC         7 non-null      object
72  Fence          281 non-null    object
73  MiscFeature    54 non-null     object
74  MiscVal        1460 non-null   int64
75  MoSold         1460 non-null   int64
76  YrSold         1460 non-null   int64
77  SaleType       1460 non-null   object
78  SaleCondition  1460 non-null   object
```

```
 79  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(34), object(43)
memory usage: 912.6+ KB
```

```
print(dataset['SalePrice'].describe())
plt.figure(figsize=(9, 8))
sns.distplot(dataset['SalePrice'], color='g', bins=100,
hist_kws={'alpha': 0.4});
```

```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

```
/tmp/ipykernel_30/3010099981.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['SalePrice'], color='g', bins=100,
hist_kws={'alpha': 0.4});
/opt/conda/lib/python3.10/site-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):
```

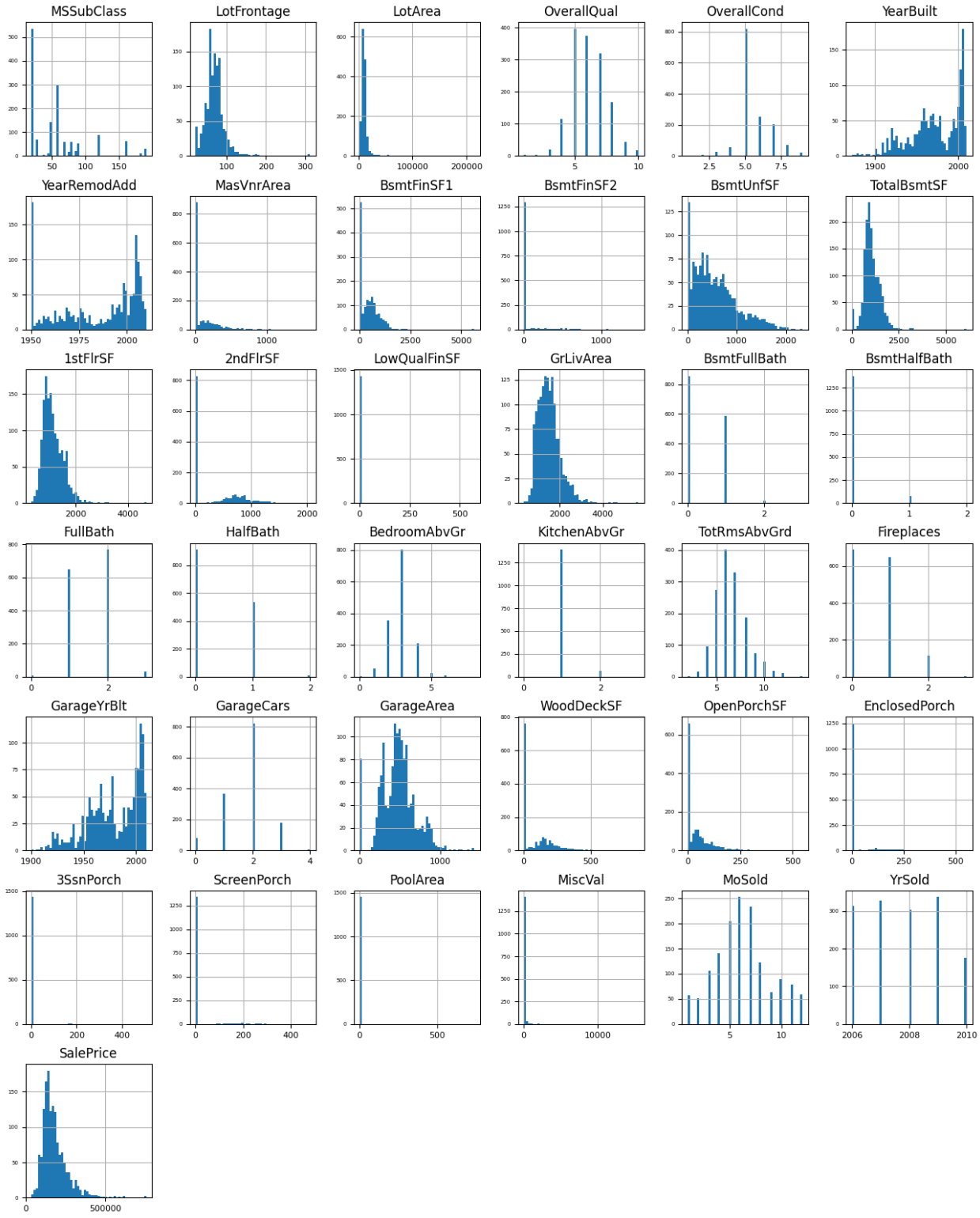



```
num = dataset.select_dtypes(include = ['float64', 'int64'])
num.head()
```

```
   MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond
YearBuilt  \
0          60         65.0     8450            7            5
2003
1          20         80.0     9600            6            8
1976
2          60         68.0    11250            7            5
2001
3          70         60.0     9550            7            5
1915
4          60         84.0    14260            8            5
2000
```

```
   YearRemodAdd  MasVnrArea  BsmtFinSF1  BsmtFinSF2  ...
WoodDeckSF  \
0          2003       196.0         706           0  ...           0

1          1976         0.0         978           0  ...         298

2          2002       162.0         486           0  ...           0

3          1970         0.0         216           0  ...           0

4          2000       350.0         655           0  ...         192


   OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea
MiscVal  \
0           61              0          0            0         0
0
1            0              0          0            0         0
0
2           42              0          0            0         0
0
3           35            272          0            0         0
0
4           84              0          0            0         0
0

   MoSold  YrSold  SalePrice
0       2    2008     208500
1       5    2007     181500
2       9    2008     223500
3       2    2006     140000
4      12    2008     250000

[5 rows x 37 columns]
```

```
num.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=5);
```




```
import numpy as np

def split_dataset(datasets, test_ratio=0.30):
```

```
  test_indices = np.random.rand(len(datasets)) < test_ratio
  return datasets[~test_indices], datasets[test_indices]

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

```
1034 examples in training, 426 examples in testing.
```

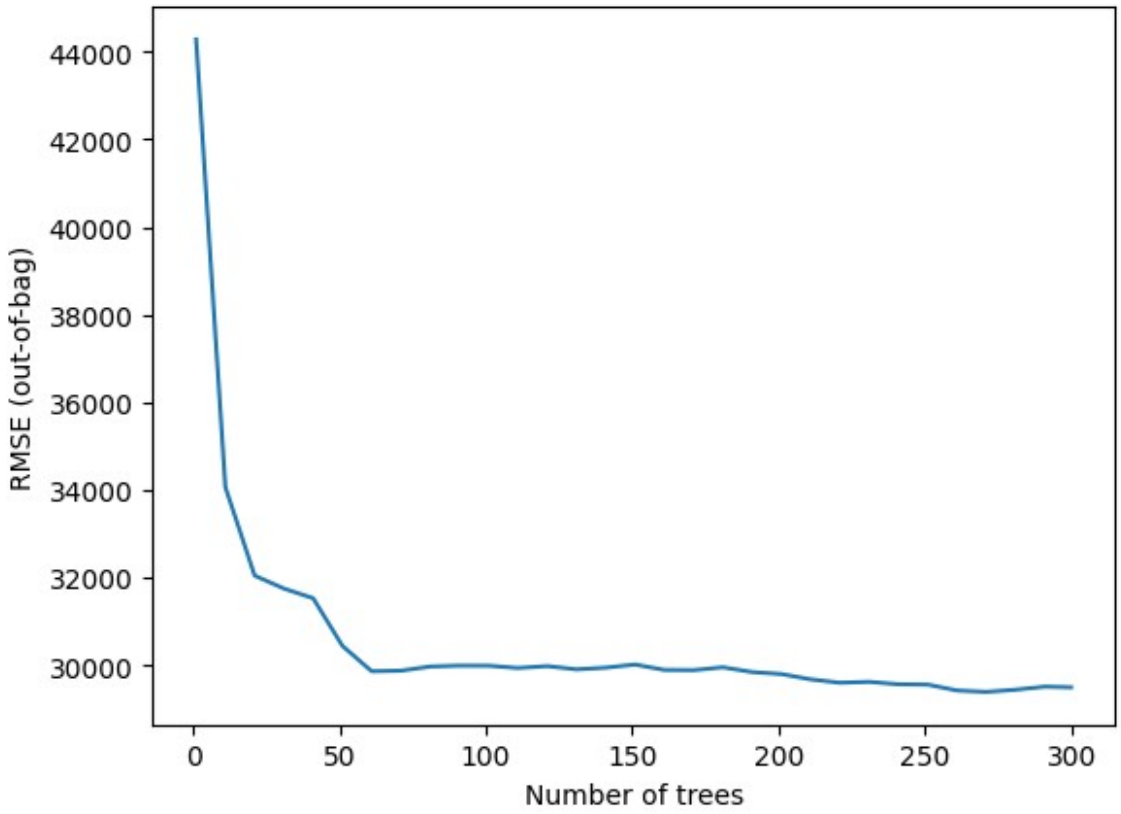
```
label = 'SalePrice'
train_ds = tfdf.keras.pd_dataframe_to_tf_dataset(train_ds_pd,
label=label, task = tfdf.keras.Task.REGRESSION)
valid_ds = tfdf.keras.pd_dataframe_to_tf_dataset(valid_ds_pd,
label=label, task = tfdf.keras.Task.REGRESSION)

rf = tfdf.keras.RandomForestModel(task = tfdf.keras.Task.REGRESSION)
```

```
Use /tmp/tmpaws02nri as temporary training directory
```

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

```
Reading training dataset...
Training dataset read in 0:00:05.892970. Found 1034 examples.
Training model...

[INFO 24-11-20 19:10:02.5240 UTC kernel.cc:1233] Loading model from
path /tmp/tmpaws02nri/model/ with prefix b7b47595902e4358

Model trained in 0:00:02.141273
Compiling model...

[INFO 24-11-20 19:10:02.9571 UTC decision_forest.cc:734] Model loaded
with 300 root(s), 98382 node(s), and 74 input feature(s).
[INFO 24-11-20 19:10:02.9572 UTC abstract_model.cc:1362] Engine
"RandomForestOptPred" built
[INFO 24-11-20 19:10:02.9573 UTC kernel.cc:1061] Use fast generic
engine

Model compiled.

<IPython.core.display.HTML object>
```

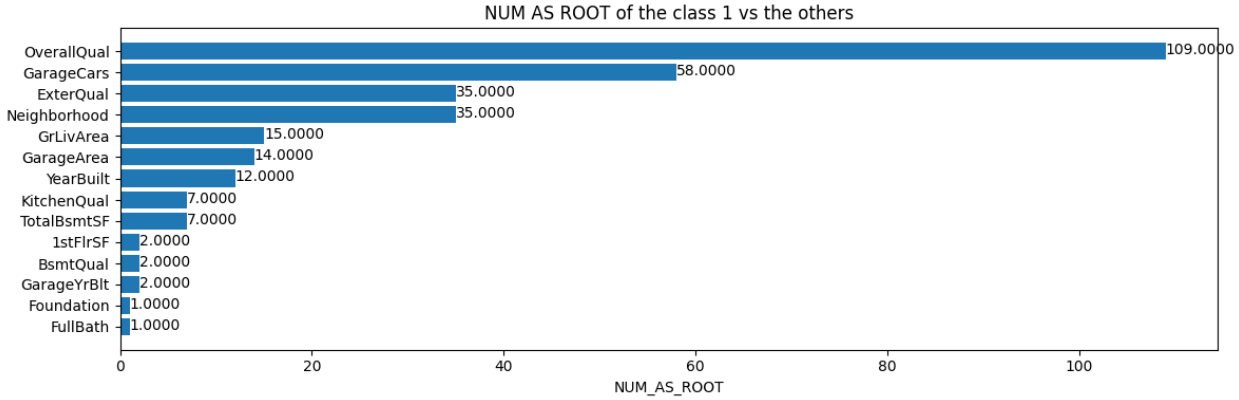
```
import matplotlib.pyplot as plt
logs = rf.make_inspector().training_logs()
plt.plot([log.num_trees for log in logs], [log.evaluation.rmse for log
in logs])
plt.xlabel("Number of trees")
plt.ylabel("RMSE (out-of-bag)")
plt.show()
```




```
plt.figure(figsize=(12, 4))

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

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

variable_importance_metric = "NUM_AS_ROOT"
variable_importances = inspector.variable_importances()
[variable_importance_metric]


feature_names = [vi[0].name for vi in variable_importances]
feature_importances = [vi[1] for vi in variable_importances]
feature_ranks = range(len(feature_names))

bar = plt.barh(feature_ranks, feature_importances, label=[str(x) for x
in feature_ranks])
plt.yticks(feature_ranks, feature_names)
plt.gca().invert_yaxis()
for importance, patch in zip(feature_importances, bar.patches):
  plt.text(patch.get_x() + patch.get_width(), patch.get_y(),
f"{importance:.4f}", va="top")
```

```
plt.xlabel(variable_importance_metric)
plt.title("NUM AS ROOT of the class 1 vs the others")
plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 3s 3s/step - loss: 0.0000e+00
loss: 0.0000
```




```
test_file_path = "/kaggle/input/dataset/test.csv"
test_data = pd.read_csv(test_file_path)
ids = test_data.pop('Id')

test_ds = tfdf.keras.pd_dataframe_to_tf_dataset(
    test_data,
    task = tfdf.keras.Task.REGRESSION)

preds = rf.predict(test_ds)
output = pd.DataFrame({'Id': ids,
                       'SalePrice': preds.squeeze()})

output.head()
```

```
2/2 [==============================] - 1s 23ms/step
```

	Id	SalePrice
0	1461	129303.335938
1	1462	154941.921875
2	1463	178009.734375
3	1464	184250.000000
4	1465	194554.796875

```
sample_submission_df =
pd.read_csv('/kaggle/input/dataset/sample_submission.csv')
sample_submission_df['SalePrice'] = rf.predict(test_ds)
sample_submission_df.to_csv('/kaggle/working/submission.csv',
```

```
index=False)
sample_submission_df.head()
```

```
2/2 [==============================] - 0s 23ms/step
```

	Id	SalePrice
0	1461	129303.335938
1	1462	154941.921875
2	1463	178009.734375
3	1464	184250.000000
4	1465	194554.796875