House Prices Prediction using TensorFlow Decision Forests

1 Introduction

Accurately predicting house prices is crucial for stakeholders in the real estate market, including buyers, sellers, and investors. Utilizing machine learning techniques, specifically TensorFlow Decision Forests (TFDF), can enhance the precision of these predictions. This report outlines the process of training a Random Forest model using TFDF on the House Prices dataset, aiming to forecast house sale prices based on various features.

Decision Forests are a family of tree-based models including Random Forests and Gradient Boosted Trees. They are the best place to start when working with tabular data, and will often outperform (or provide a strong baseline) before you begin experimenting with neural networks.

2 Data

2.1 Dataset

The data is composed of 81 columns and 1460 entries.

	id	MSSubClass	MSZoning	LotFrontage	…	SaleType	SaleCondition	SalePrice
0	1	60	RL	65.0	…	WD	Normal	208500
1	2	20	RL	80.0	…	WD	Normal	181500
2	3	60	RL	68.0	…	WD	Normal	223500

Table [1]: Train Dataset (3 rows × 81 columns) , first 3 entries and 79 features in addition of id.

2.2 Features

There are 79 feature columns. Using these features the model has to predict the house sale price indicated by the label column named SalePrice.

The features are as follows:

#	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	1452 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

2.3 Distribution

2.3.1 House Price Distribution

Now let us take a look at how the house prices are distributed.

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

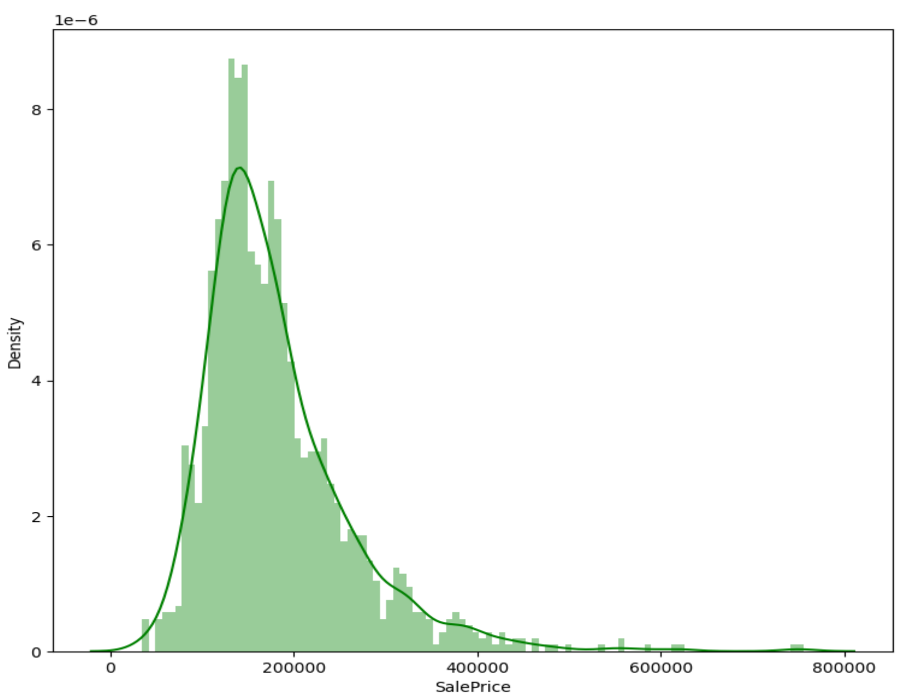

Figure [1]: House Price Distribution, which indicates that the price of a large number of houses are around 200000 and a few of them seem to be too much more expensive (Their prices are from 400000 increased to around 800000) , which obviously increases the Mean value.

2.3.2 Numerical data distribution

We will now take a look at how the numerical features are distributed. In order to do this, let us first list all the types of data from our dataset and select only the numerical ones:

[dtype('O'), dtype('int64'), dtype('float64')]

	MSSubClass	LotFrontage	LotArea	...	MoSold	YrSold	SalePrice
0	60	65.0	8450	...	2	2008	208500
1	20	80.0	9600	...	5	2007	181500
2	60	68.0	11250	...	9	2008	223500
3	70	60.0	9550	...	2	2006	140000
4	60	84.0	14260	...	12	2008	250000

Table [2]: (5 rows × 37 columns) , entries with int64 and float64 datatype

Now let us plot the distribution for all the numerical features.

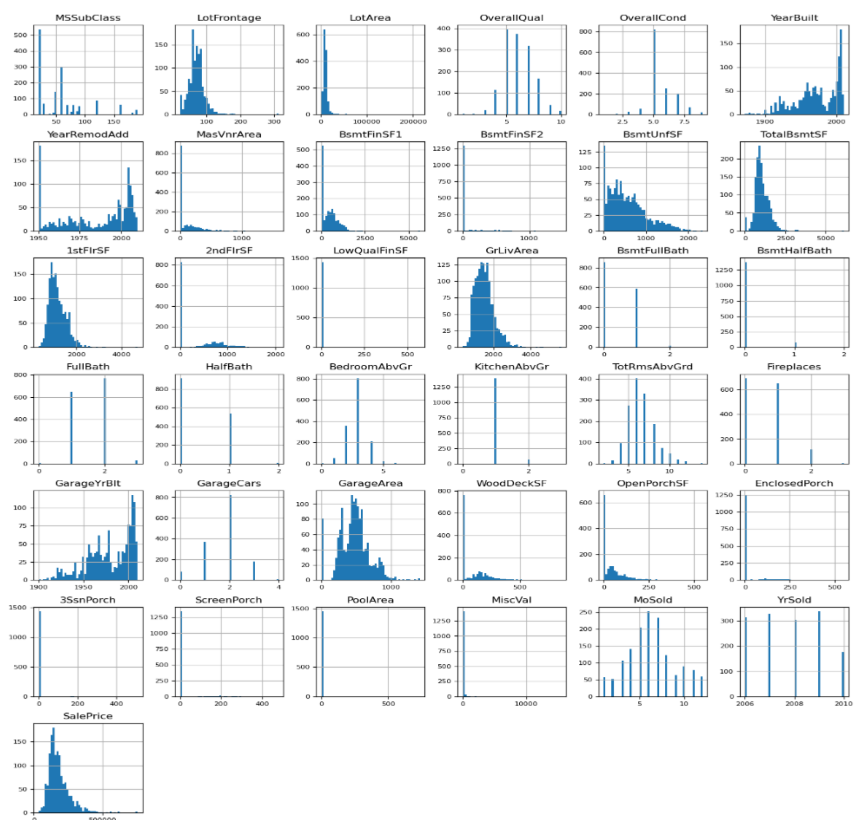

Figure [2]: the distribution for all the numerical features, which help with the capability of new pattern identification as well as ease the analysis process due to more definite information related to each feature individually

2.4 Prepare the dataset

This dataset contains a mix of numeric, categorical and missing features. TF-DF supports all these feature types natively, and no preprocessing is required. This is one advantage of tree-based models, making them a great entry point to Tensorflow and ML.

We split the dataset into training and testing datasets:

1010 examples in training, 450 examples in testing

There's one more step required before we can train the model. We need to convert the datatset from Pandas format (pd.DataFrame) into TensorFlow Datasets format (tf.data.Dataset).

[TensorFlow Datasets] is a high-performance data loading library which is helpful when training neural networks with accelerators like GPUs and TPUs.

3 Models

There are several tree-based models to choose from.

- RandomForestModel
- GradientBoostedTreesModel
- CartModel
- DistributedGradientBoostedTreesModel

The list of the all the available models in TensorFlow Decision Forests:

tensorflow_decision_forests.keras.RandomForestModel,

tensorflow_decision_forests.keras.GradientBoostedTreesModel,

tensorflow_decision_forests.keras.CartModel,

tensorflow_decision_forests.keras.DistributedGradientBoostedTreesModel

3.1 Random Forest

To start, we'll work with a Random Forest. This is the most well-known of the Decision Forest training algorithms.

A Random Forest is a collection of decision trees, each trained independently on a random subset of the training dataset (sampled with replacement). The algorithm is unique in that it is robust to overfitting, and easy to use.

A Random Forest model was chosen due to its robustness in handling diverse datasets and its capability to model complex interactions between features. The TFDF library in TensorFlow was utilized to construct this ensemble learning model, which builds multiple decision trees and aggregates their outputs for improved predictive performance.

3.2 Visualize the Model

One benefit of tree-based models is that it can be easily visualized. The default number of trees used in the Random Forests is 300.

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

Before training the dataset we have manually separated 20% of the dataset for validation.

We can also use Out of bag (OOB) score to validate our RandomForestModel. To train a Random Forest Model, a set of random samples from training set are choosen by the algorithm and the rest of the samples are used to finetune the model. The subset of data that is not chosen is known as Out of bag data (OOB). OOB score is computed on the OOB data.

The training logs show the Root Mean Squared Error (RMSE) evaluated on the out-of-bag dataset according to the number of trees in the model.

Note: Smaller values are better for this hyperparameter.

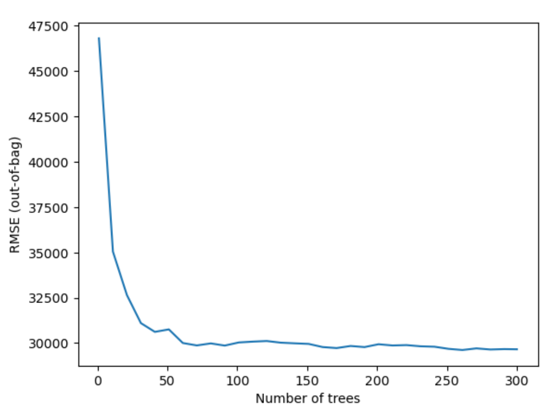

Figure [3]: Evaluate the model on the Out of bag (OOB) data and the validation dataset. What is pretty comprehensible from this plot is that when the number of trees increases we can observe that the RMSE index decreases.

4 Analysis

4.1 Variable importances

Variable importances generally indicate how much a feature contributes to the model predictions or quality. There are several ways to identify important features using TensorFlow Decision Forests. Let us list the available Variable Importances for Decision Trees:

Available variable importances:

INV_MEAN_MIN_DEPTH

NUM_AS_ROOT

NUM_NODES

SUM_SCORE

As an example, let us display the important features for the Variable Importance NUM_AS_ROOT.

The larger the importance score for NUM_AS_ROOT, the more impact it has on the outcome of the model.

By default, the list is sorted from the most important to the least. From the output you can infer that the feature at the top of the list is used as the root node in most number of trees in the random forest than any other feature.

("OverallQual" (1; #62), 121.0),

("GarageCars" (1; #32), 49.0),

("ExterQual" (4; #22), 40.0),

("Neighborhood" (4; #59), 35.0),

("GrLivArea" (1; #38), 21.0),

("GarageArea" (1; #31), 15.0),

("BsmtQual" (4; #14), 7.0),

("YearBuilt" (1; #76), 5.0),

("KitchenQual" (4; #44), 4.0),

("TotalBsmtSF" (1; #73), 3.0)

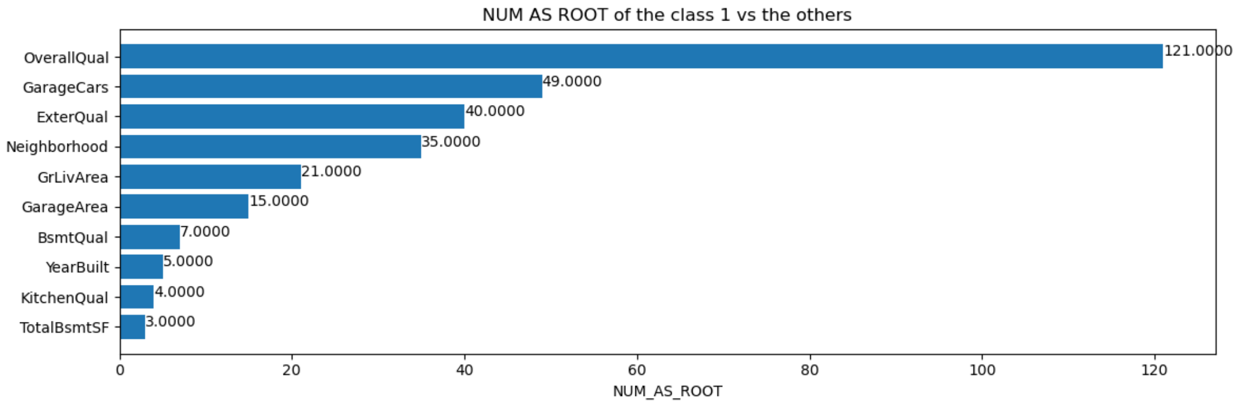

Figure [4]: the variable importances from the inspector using Matplotlib in which, the top 10 most important and most effective features can be observed.

4 Result

Finally predict on the competition test data using the model.

	id	SalePrice
0	1461	123554.718750
1	1462	153939.062500
2	1463	176793.765625
3	1464	183828.296875
4	1465	193644.484375

Table [3]: result of prediction on the competition test data using the model

Conclusion

Implementing a Random Forest model using TensorFlow Decision Forests provides a reliable approach to predicting house prices based on a wide array of features. The model demonstrates strong performance, capturing significant variance in the data and offering accurate price estimations. Future enhancements could involve incorporating additional features, exploring other ensemble methods, or further fine-tuning hyperparameters to improve predictive accuracy.

References

[1] https://www.kaggle.com/code/gusthema/house-prices-prediction-using-tfdf#House-Prices-Prediction-using-TensorFlow-Decision-Forests

Maryam Soleimani 401222075