Machine learning on house price prediction using the distributed computing system

Abstract
Machine learning is widely used in house price prediction. However, the rapid expansion in data size challenges the computing power on which machine learning algorithms is run. This project utilised scalable machine learning models within the Apache Spark machine learning library running on top of the Hadoop Distributed File System (HDFS). The volume of data could be easily scaled up to big data by using such technics.

Introduction and literature

House price prediction is important to property agents and investors. Many studies on house price prediction focus on comparing the prediction performance of different models without discussion of rapid growing data size, such as (Ho, Tang and Wong, 2021). At the same time, other studies focused on different house price predictions of different locations, such as (Phan, 2018). However, the rapid expansion of big data challenges the computing power where such algorithms are run. For example, Johnson (2021) reported that around 306.4 billion emails could be sent and received every day globally in 2020. Few studies were found to address such challenges. On the other side, distributed computing has developed rapidly since the initial release of HDFS. Especially the release of Apache Spark, which contains machine learning packages, could easily scale up data analysis. This project was inspired to deal with the challenges brought by big data via using Apache Spark and HDFS technologies in house price predictions.

Aim

As above stated, the rapid expansion of data challenges the computing power on machine-learning house price prediction. This project aims to deploy machine learning with distributed computing systems by quantitative approach by a case study of Melbourne house prices.

Objectives include,

Firstly, examine the recent literature to understand the state of the art of the topic area and identify the gaps that this project could contribute by reviewing reputable science journals and news articles in the initial phase of the project.

Secondly, collect data through the Kaggle platform and prepare data for regression analysis on the distributed computing system (pyspark).

Thirdly, build Random Forest Regression (RFR) model for house price prediction utilising Pyspark machine learning library(MLlib) in the project implementation phase on HDFS platform.

After that, optimise the RFR model by eliminating the insignificant features to improve overall computing efficiency after the initial model evaluation.

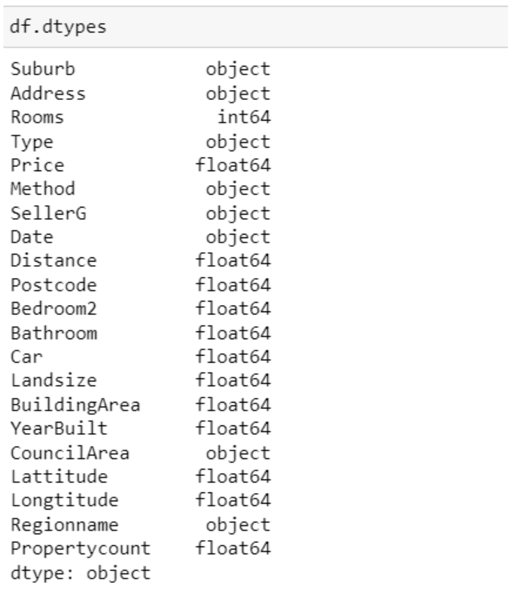
Subsequently, build a Decision Tree Regression (DTR) model to compare models' performance.

Finally, evaluate model performance by metrics (rmse & mse) to gain insights on house price and validate the Pyspark approach influence on machine learning models' performance at the end of the project.

Data (Please refer to "Melbourn_housing_FULL.csv" and "Melbourne_housing_FULL_DataCleaning.ipynb")

Melbourne is one of the most populous cities in Australia. The Melbourne housing is published under an open-source license (Pino, 2018). Two datasets are available on the website, which were scraped from publicly available results and the dataset, "Melbourne_housing_FULL.csv", was selected for this project as it contains more features compared to the other one (Pino, 2018).

There are 34,857 observations and 21 variables in the dataset, with 13 numerical variables and eight categorical variables shown as below,



```
df.dtypes
```

```
Suburb            object
Address           object
Rooms              int64
Type              object
Price            float64
Method            object
SellerG           object
Date              object
Distance         float64
Postcode         float64
Bedroom2         float64
Bathroom         float64
Car              float64
Landsize         float64
BuildingArea     float64
YearBuilt        float64
CouncilArea       object
Lattitude        float64
Longtitude       float64
Regionname        object
Propertycount    float64
dtype: object
```

Each observation represents a real sales record from 2016 to 2018, and each variable represents a feature. While columns are self-explanatory to some extent, more details of these columns could be explained from the Kaggle website(Pino, 2018). These variables could be grouped into three categories as below,

Locations including 'Suburb', 'Address', 'Distance', 'Postcode', 'CouncilArea', 'Lattitude', 'Longtitude' and 'Regionname'

House features including 'Rooms', 'Type', 'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'BuildingArea' and 'YearBuilt'

Transaction features including 'Price', 'Method', 'SellerG', 'Date' and 'Propertycount'

The dependent variable is house price, while other variables are independent variables.

Original Data preview

The preview of original data is to identify the characteristics and patterns within the data and provide directions for data pre-processing for the house price analysis.

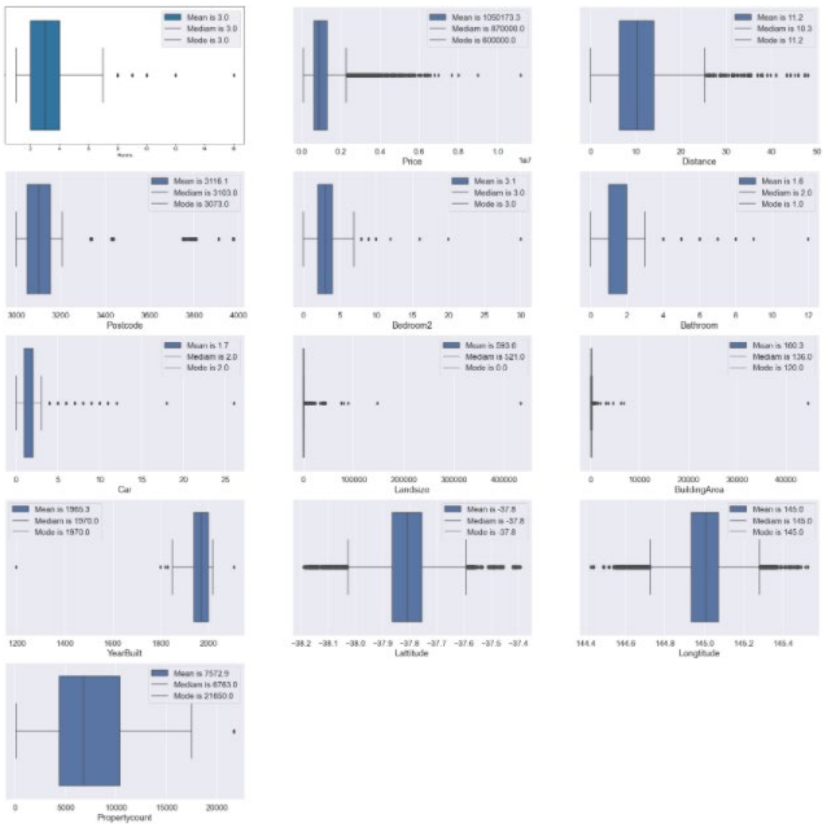
Firstly, missing values were checked as follows. Features “BuildingArea” and “YearBuilt” contains more than 50% missing values.

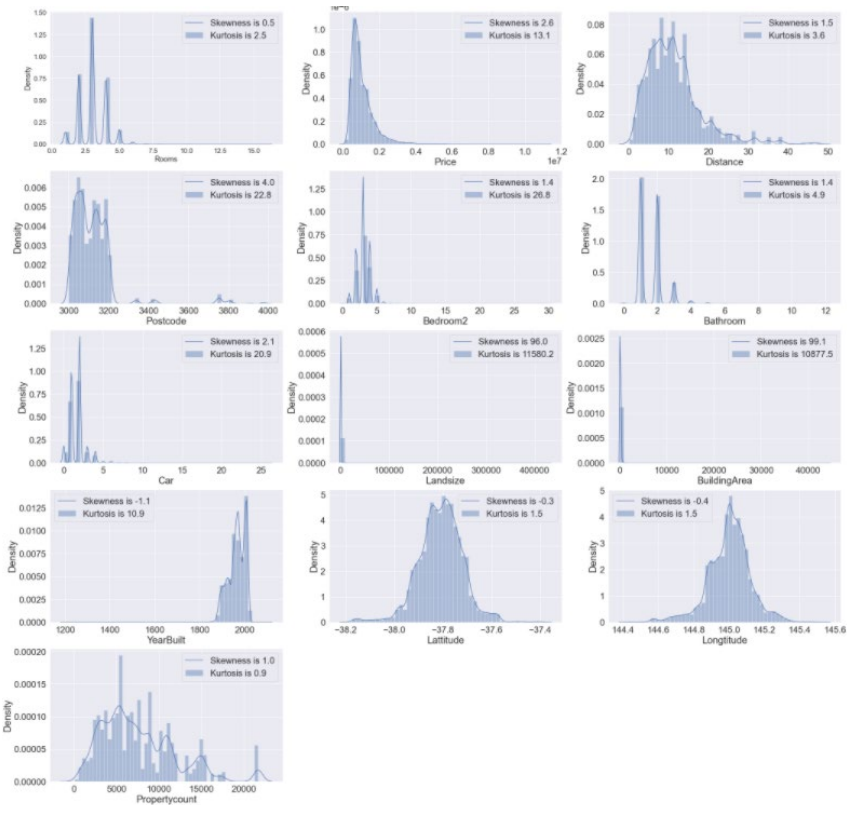
```python
for col in df.columns:
    Nullcounts=df[col].isna().sum()
    percent=(Nullcounts/len(df[col]))*100
    print('Null values for {}'.format(col) + ' is: {}'.format(Nullcounts) + "  Percentage: {:.1f}%".format(percent))
```

```
Null values for Suburb is: 0  Percentage: 0.0%
Null values for Address is: 0  Percentage: 0.0%
Null values for Rooms is: 0  Percentage: 0.0%
Null values for Type is: 0  Percentage: 0.0%
Null values for Price is: 7610  Percentage: 21.8%
Null values for Method is: 0  Percentage: 0.0%
Null values for SellerG is: 0  Percentage: 0.0%
Null values for Date is: 0  Percentage: 0.0%
Null values for Distance is: 1  Percentage: 0.0%
Null values for Postcode is: 1  Percentage: 0.0%
Null values for Bedroom2 is: 8217  Percentage: 23.6%
Null values for Bathroom is: 8226  Percentage: 23.6%
Null values for Car is: 8728  Percentage: 25.0%
Null values for Landsize is: 11810  Percentage: 33.9%
Null values for BuildingArea is: 21115  Percentage: 60.6%
Null values for YearBuilt is: 19306  Percentage: 55.4%
Null values for CouncilArea is: 3  Percentage: 0.0%
Null values for Lattitude is: 7976  Percentage: 22.9%
Null values for Longtitude is: 7976  Percentage: 22.9%
Null values for Regionname is: 3  Percentage: 0.0%
Null values for Propertycount is: 3  Percentage: 0.0%
```

Secondly, feature range and distribution statistics are as below. Some extreme values highlighted in row "max" are far from the interquartile range, probably outliers. Besides, "Landsize" and "BuildingArea" are heavily skewed. Moreover, the observations of land size value of zero are outliers

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Lattitude	Longtitude	Propertycount
count	34,857	27,247	34,856	34,856	26,640	26,631	26,129	23,047	13,742	15,551	26,881	26,881	34,854
mean	3.0	1,050,173.0	11.2	3,116.1	3.1	1.6	1.7	593.6	160.3	1,965.3	-37.8	145.0	7,572.9
std	1.0	641,467.1	6.8	109.0	1.0	0.7	1.0	3,398.8	401.3	37.3	0.1	0.1	4,428.1
min	1.0	85,000.0	0.0	3,000.0	0.0	0.0	0.0	0.0	0.0	1,196.0	-38.2	144.4	83.0
25%	2.0	635,000.0	6.4	3,051.0	2.0	1.0	1.0	224.0	102.0	1,940.0	-37.9	144.9	4,385.0
50%	3.0	870,000.0	10.3	3,103.0	3.0	2.0	2.0	521.0	136.0	1,970.0	-37.8	145.0	6,763.0
75%	4.0	1,295,000.0	14.0	3,156.0	4.0	2.0	2.0	670.0	188.0	2,000.0	-37.8	145.1	10,412.0
max	16.0	11,200,000.0	48.1	3,978.0	30.0	12.0	26.0	433,014.0	44,515.0	2,106.0	-37.4	145.5	21,650.0
Median	3.0	870,000.0	10.3	3,103.0	3.0	2.0	2.0	521.0	136.0	1,970.0	-37.8	145.0	6,763.0
Mode	3.0	600,000.0	11.2	3,073.0	3.0	1.0	2.0	0.0	120.0	1,970.0	-37.8	145.0	21,650.0
Skewness	0.5	2.6	1.5	4.0	1.4	1.4	2.1	90.6	99.1	-1.1	-0.3	-0.4	1.0
Kurtosis	2.5	13.1	3.6	22.8	26.8	4.9	20.9	11,580.2	10,877.6	10.9	1.5	1.5	0.9
missing value (counts)	0	7610	1	1	8217	8226	8728	11810	21115	19306	7976	7976	3

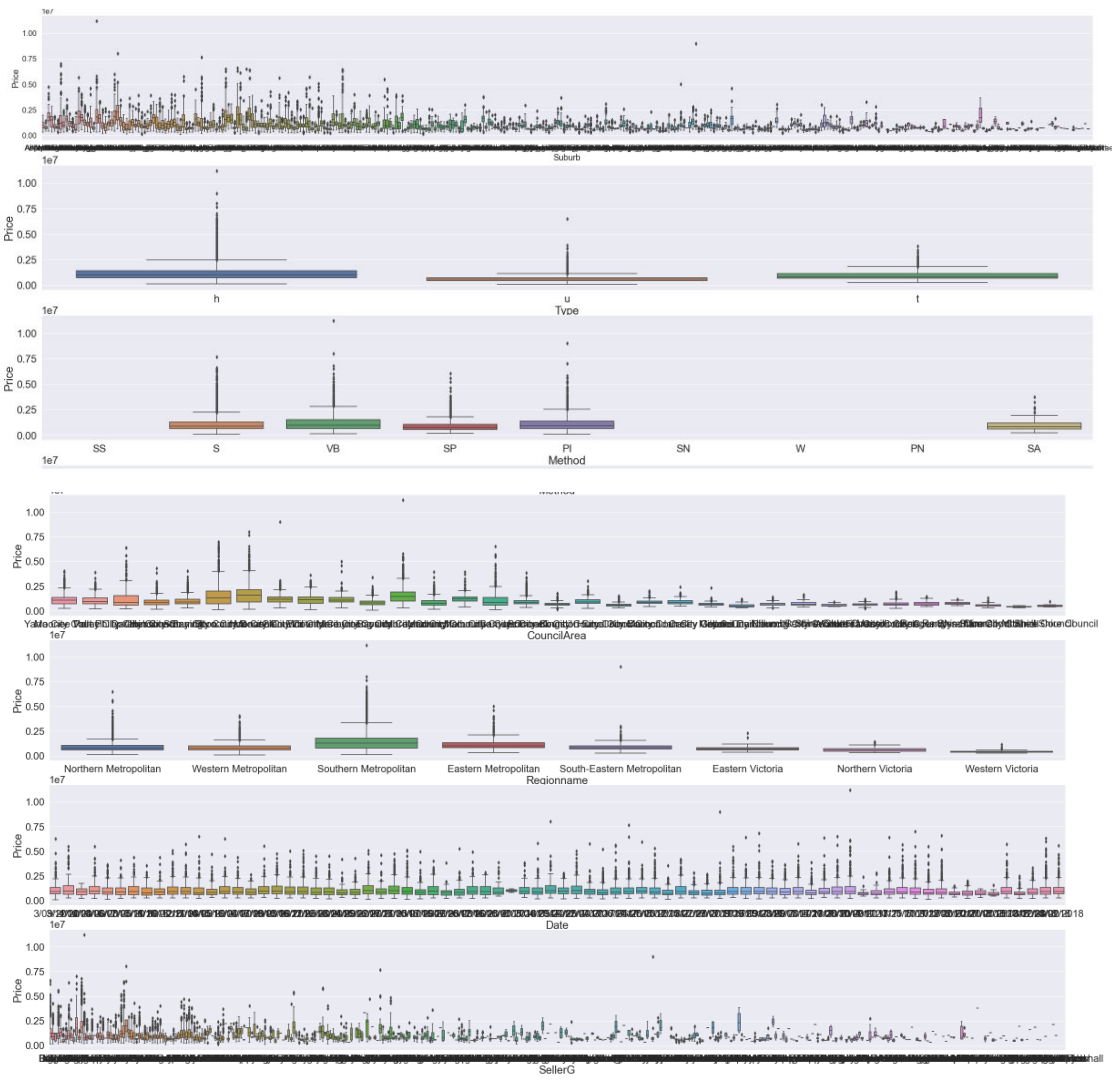




Thirdly, there are eight categorical features and their unique values are as shown below. Categorical features including 'Type', 'Method', 'Date' appeared constant to price.

```
df[categorical_features].nunique()
```

```
Suburb            351
Type                3
Method              9
CouncilArea        33
Regionname          8
Date               78
Address         34009
SellerG           388
dtype: int64
```



Data Pre-processing

The exploration from data original data suggests the following areas of pre-processing,

Removing address columns including 'Address', 'Postcode', 'Suburb', 'Regionname','CouncilArea' as these are better expressed by continuous data 'Lattitude','Longtitude'.

Removing columns of 'BuildingArea', 'YearBuilt' as the missing value is more than 50%, and it is hard to impute values with acceptable accuracy.

Removing columns of 'Type', 'Method', 'Date' as these features have a little variance to the outcome variable.

Removing outliers of 'Landsize' equals zero, as property land size could not be zero.

Removing the 0.5% of extreme values in columns 'Rooms', 'Bedroom2', 'Bathroom', 'Car', 'Landsize' as such values do not sound reasonable. Besides, dropping these data points will cause little information loss

Removing missing values in price as we do not want to increase bias in outcome variable by imputing it. Also, the dataset is big enough to exclude such observations

Imputing missing value with median values for numerical variables as mean value is sensitive to extreme values and with mode value for categorical variables.

Check and drop duplicates.

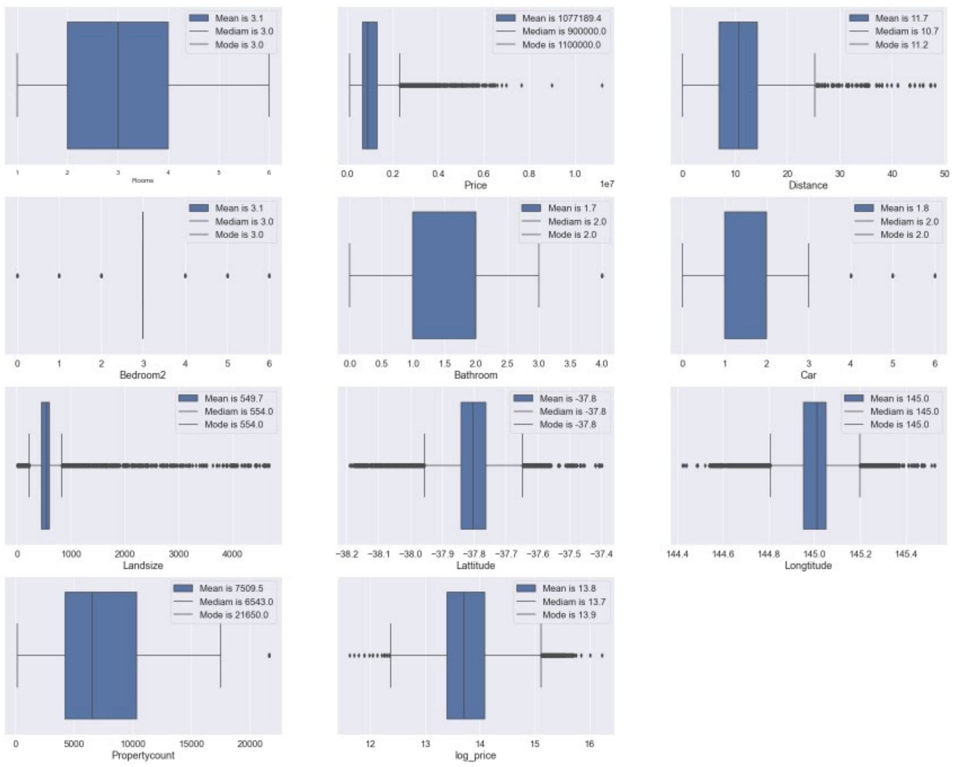
Logarithm transformation on price. Transforming outcome variable to normal distribution allows better visualisation of feature associations.

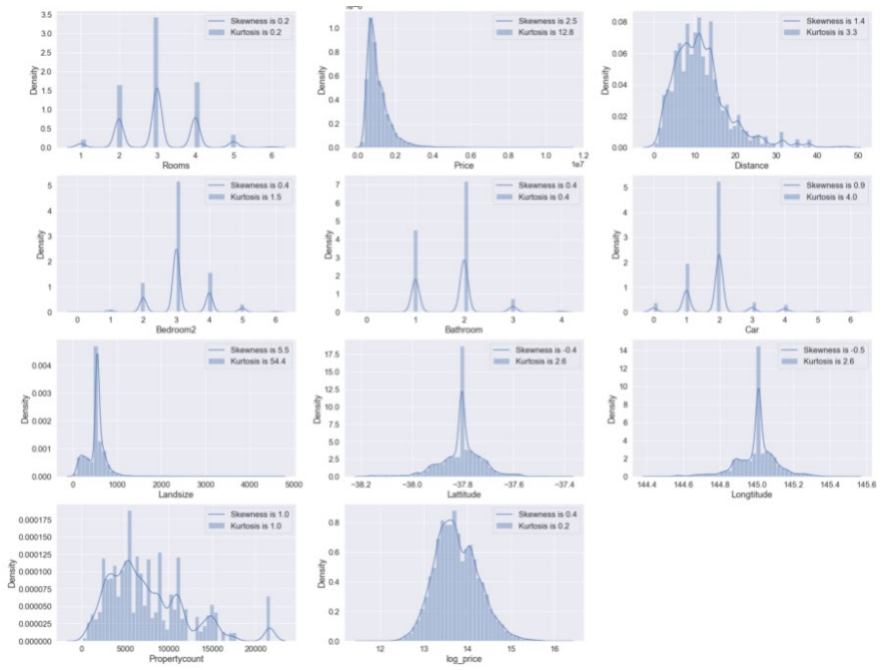
Final data summary

The final data has 24,989 observations and 11 numerical variables and only one categorical of “SellerG” containing 337 unique values. The mean, median, min, max values and standard deviations are listed as below,

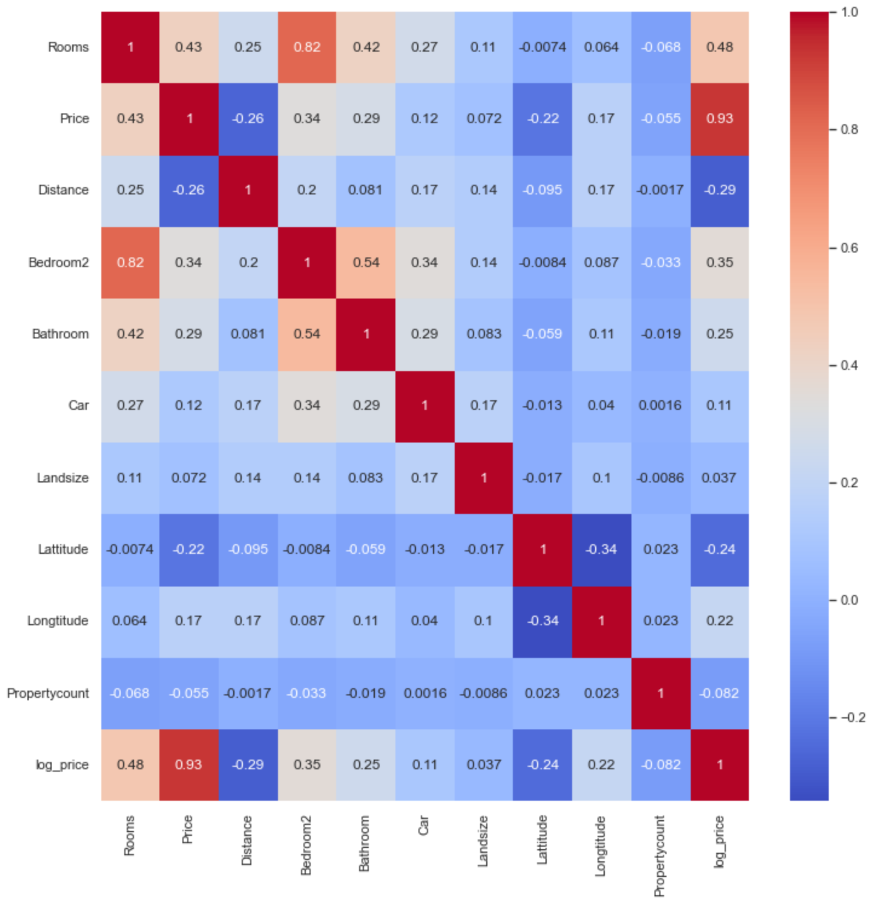
	Rooms	Price	Distance	Bedroom2	Bathroom	Car	Landsize	Lattitude	Longtitude	Propertycount	log_price
count	24,989	24,989	24,989	24,989	24,989	24,989	24,989	24,989	24,989	24,989	24,989
mean	3.1	1,077,189.0	11.7	3.1	1.7	1.8	549.7	-37.8	145.0	7,509.5	13.8
std	0.9	638,670.2	6.8	0.7	0.6	0.8	316.5	0.1	0.1	4,509.0	0.5
min	1.0	112,000.0	0.0	0.0	0.0	0.0	1.0	-38.2	144.4	121.0	11.6
25%	2.0	660,000.0	7.0	3.0	1.0	1.0	453.0	-37.8	145.0	4,217.0	13.4
50%	3.0	900,000.0	10.7	3.0	2.0	2.0	554.0	-37.8	145.0	6,543.0	13.7
75%	4.0	1,315,000.0	14.3	3.0	2.0	2.0	603.0	-37.8	145.1	10,331.0	14.1
max	6.0	11,200,000.0	48.1	6.0	4.0	6.0	4,679.0	-37.4	145.5	21,650.0	16.2
Median	3.0	900,000.0	10.7	3.0	2.0	2.0	554.0	-37.8	145.0	6,543.0	13.7
Mode	3.0	1,100,000.0	11.2	3.0	2.0	2.0	554.0	-37.8	145.0	21,650.0	13.9
Skewness	0.2	2.5	1.4	0.4	0.4	0.9	5.5	-0.4	-0.5	1.0	0.4
Kurtosis	0.2	12.8	3.3	1.5	0.4	4.0	54.4	2.6	2.6	1.0	0.2
missing value (counts)	0	0	0	0	0	0	0	0	0	0	0

As shown below, these numerical features are more normally distributed with less extreme values comparing original data. Extreme values can be seen from most variables.



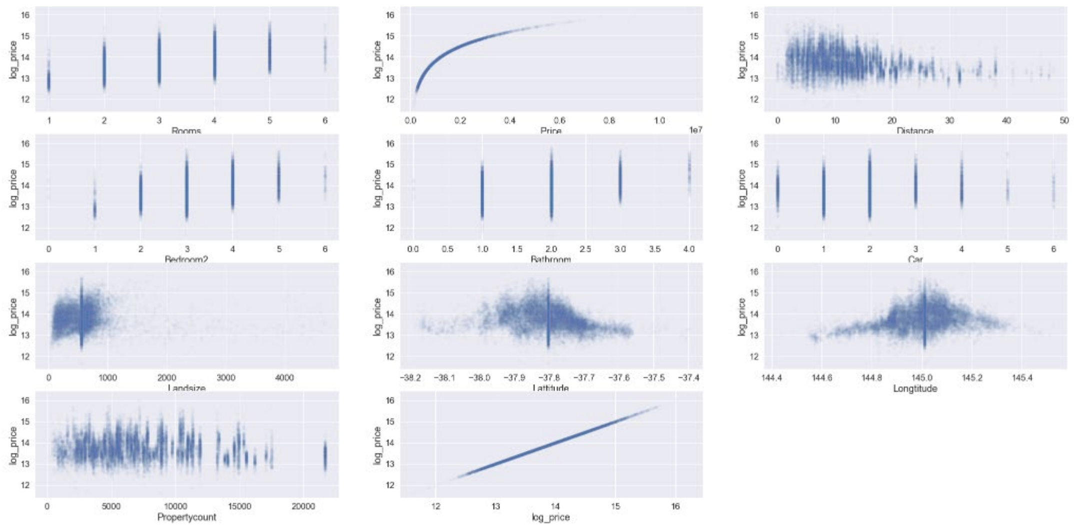


There are some strong correlations among house features “Bedroom2”, “Bathroom” and “Rooms” as shown below,



	Rooms	Price	Distance	Bedroom2	Bathroom	Car	Landsize	Lattitude	Longtitude	Propertycount	log_price
Rooms	1	0.43	0.25	0.82	0.42	0.27	0.11	-0.0074	0.064	-0.068	0.48
Price	0.43	1	-0.26	0.34	0.29	0.12	0.072	-0.22	0.17	-0.055	0.93
Distance	0.25	-0.26	1	0.2	0.081	0.17	0.14	-0.095	0.17	-0.0017	-0.29
Bedroom2	0.82	0.34	0.2	1	0.54	0.34	0.14	-0.0084	0.087	-0.033	0.35
Bathroom	0.42	0.29	0.081	0.54	1	0.29	0.083	-0.059	0.11	-0.019	0.25
Car	0.27	0.12	0.17	0.34	0.29	1	0.17	-0.013	0.04	0.0016	0.11
Landsize	0.11	0.072	0.14	0.14	0.083	0.17	1	-0.017	0.1	-0.0086	0.037
Lattitude	-0.0074	-0.22	-0.095	-0.0084	-0.059	-0.013	-0.017	1	-0.34	0.023	-0.24
Longtitude	0.064	0.17	0.17	0.087	0.11	0.04	0.1	-0.34	1	0.023	0.22
Propertycount	-0.068	-0.055	-0.0017	-0.033	-0.019	0.0016	-0.0086	0.023	0.023	1	-0.082
log_price	0.48	0.93	-0.29	0.35	0.25	0.11	0.037	-0.24	0.22	-0.082	1

Independent variables are associated to the outcome variable, as shown below.



The graphs suggest poor linearity between independent variables and dependent variable.

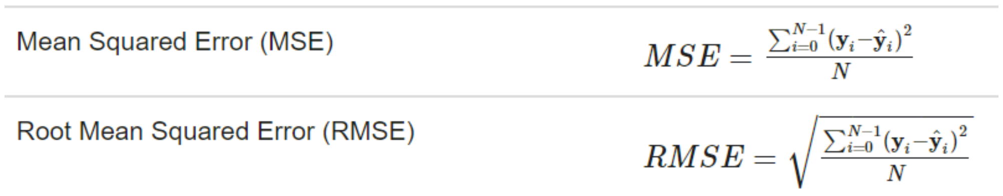
This project aims to explore machine learning models in house price prediction to the challenge of big data. The above data exploration suggests that tree-based machine learning models could be suitable for this project due to the limited project size and also the robustness of tree-based models to extreme values, correlations or lack of linearities.

Methodology and technics:
Rapid data expansion challenges the computing power for machine learning algorithms in house price prediction. This project suggested machine learning models based on the distributed computing systems by quantitative approach by a case study of the Melbourne house price prediction. The planned analysis includes two parts. The first part included univariate analysis, independent variables association with outcome variable analysis, and multivariate association analysis on original data with Python. The second part included house price prediction regression analysis with pyspark.

Analysis planned in two parts and documented in two Jupiter notebooks, Real transaction data on house price is usually messy. Examining and pre-processing raw data is essential to good house price analysis. Moreover, the exploration could decide the appropriate models for analysis.

- Examine literature and collect raw data
- Investigate raw data and pre-process it from missing values, outliers, extreme values etc.
- Probe data and form summary statistics.
- Upload data onto HDFS, where the data will be split and stored parallelly across a cluster of computers denoted as worker nodes.
- Read the data from HDFS into Pyspark application which lays on top of the HDFS system.
  " Apache Spark is the Smartphone of Big Data" (Gutierrez, 2015). Spark was an integrated platform with its core covering a wide range of APIs such as Python and supporting reading data from a variety of data warehouses such as HDFS. Moreover, Spark supports a wide range of applications such as machine learning libraries, making it flexible in a combination of different computer languages and applications(Gutierrez, 2015). Spark's fundamental data structure is the Resilient Distributed Dataset (RDD) which is an immutable dataset and could be partitioned across multiple data nodes. RDD allows in-memory computing, lazy evaluation and fault tolerance (Drabas and Lee, 2017). Hence, data could be easily scaled up using the Spark application.
- Build a random forest regression model for house price prediction via the Pyspark machine learning library.
  The machine learning library in Spark consists of three classes, transformer (to create new data frame), estimator (to make prediction or classification) and pipeline (to chain multiple stages) (Drabas and Lee, 2017). This project will apply all these three classes in model building.
  An important aspect of model building is to decide the hyperparameters for the model. This project will use cross-validation to tune the model and find the optimised hyperparameters. As cross-validation is usually expensive, further exploration will be conducted on optimising the cross-validation pipeline model.
- Optimise random forest model and evaluate the performance of the optimised model

- Build a decision tree regression model for house price prediction.
- Evaluate different metrics (RMSE & MSE) across different models
  The Mean Squared Error (MSE) is the average of the squares of the difference between predicted values and observed values (alias, errors), while the root-mean-square error (RMSE) measures the quadratic mean of errors.



Mean Squared Error (MSE)	$MSE = \frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{N}$
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{N}}$

  Sourced from https://spark.apache.org/docs/1.6.3/mllib-evaluation-metrics.html#regression-model-evaluation

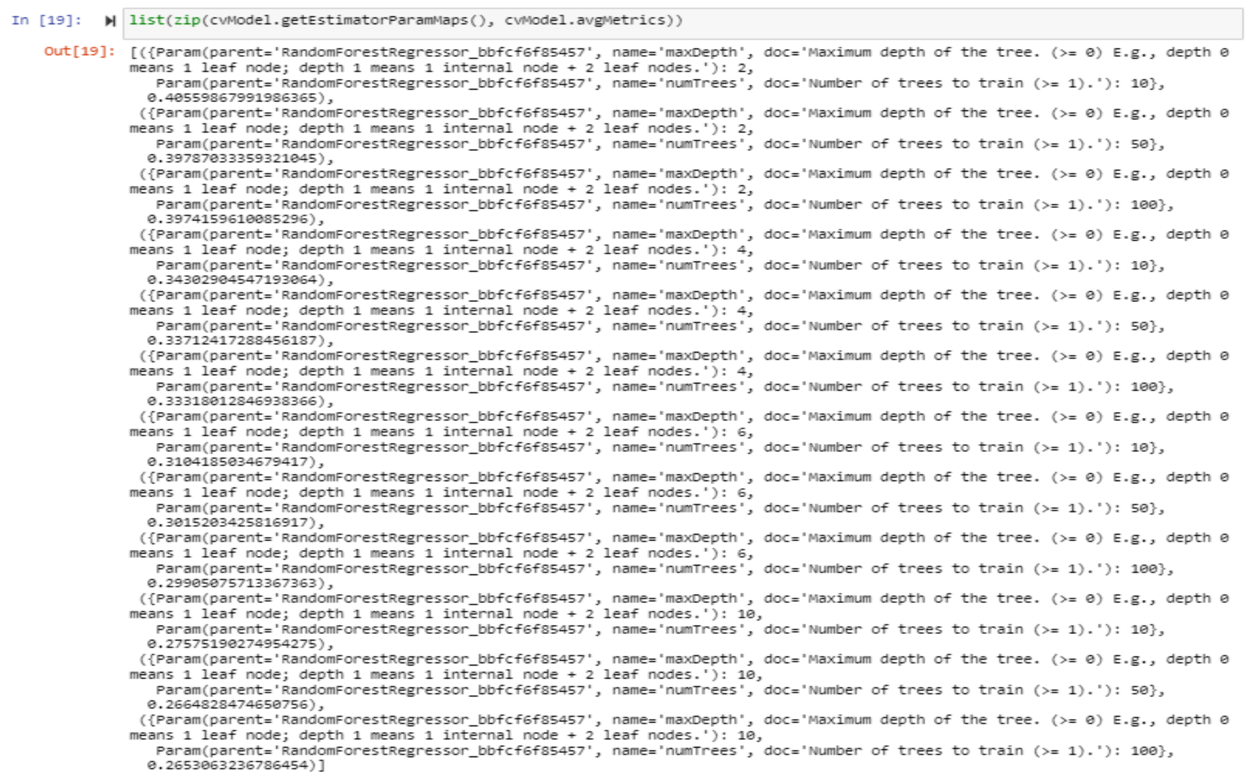
This project aims to predict house prices using distributed machine models. The exploration on raw data showed a lack of linearity between independent variables and the outcome variable. To avoid a shift of focus from distributed computing to statistic assumptions exploration and to meet project report length requirements, this study deployed tree-based models which perform properly regardless of extreme values, correlations, and lack of linearity. This project compared tree-based models in house price prediction using distributed computing systems to deal with the challenges of big data.

Implementation: (Documented as DSM010_CW2_ML.ipynb)

Summary and conclusions:
There are many studies on machine learning house price prediction. The distributed computing system, such as Pyspark and HDFS, allow the scaling up of data on house price predictions. This project used a case study of Melbourne house price prediction with the findings as below.

There were extreme values in most variables. Independent variables showed poor linearity against outcome variables. Variables “Rooms”, “Bedroom2” and “Bathroom” are highly correlated to each other. Random forest and decision tree regression models were selected for regression analysis as they are robust to extreme values, correlations and not affected by linearity.

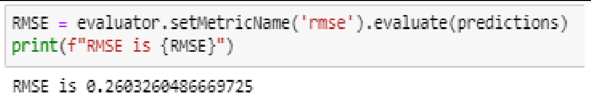
RFR hyperparameters tuning showed that the maximum depth of trees was ten, and the maximum number of trees was 100 based on metric rmse. Decision trees could consistently learn the pattern of data until tree depth reached 10, where the trees may have a minimum improvement in learning the pattern but instead memorising the data. When the number of trees increased, the ensembled decisions had a smaller variance to actual house price until the tree number reached 100, where the improvement in decision became minimum but consumed more computing resources.



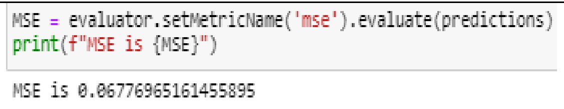
```
In [19]:  list(zip(cvModel.getEstimatorParamMaps(), cvModel.avgMetrics))

Out[19]: [({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 10},
           0.4055986799198636 5),
```

```
Out[19]: [({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 10},
           0.40559867991986365),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 50},
           0.39787033359321045),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 100},
           0.3974159610085296),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 4,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 10},
           0.34302904547193064),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 4,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 50},
           0.33712417288456187),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 4,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 100},
           0.33318012846938366),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 10},
           0.3104185034679417),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 50},
           0.3015203425816917),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 100},
           0.29905075713367363),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 10,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 10},
           0.27575190274954275),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 10,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 50},
           0.2664828474650756),
          ({Param(parent='RandomForestRegressor_bbfcf6f85457', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
         means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 10,
            Param(parent='RandomForestRegressor_bbfcf6f85457', name='numTrees', doc='Number of trees to train (>= 1).'): 100},
           0.2653063236786454)]
```

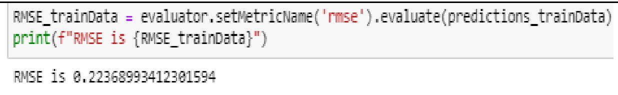
RFR model prediction performance showed small variance between training dataset and testing dataset on both metrics as below.



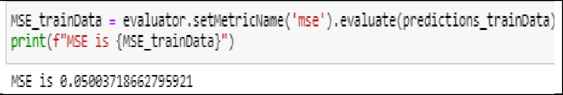
```
RMSE = evaluator.setMetricName('rmse').evaluate(predictions)
print(f"RMSE is {RMSE}")

RMSE is 0.2603260486669725
```



```
MSE = evaluator.setMetricName('mse').evaluate(predictions)
print(f"MSE is {MSE}")

MSE is 0.06776965161455895
```

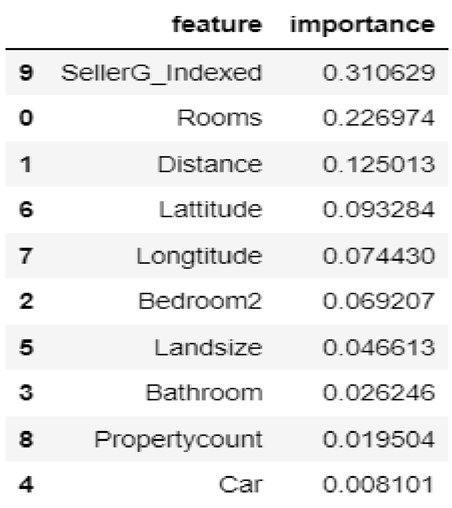


```
RMSE_trainData = evaluator.setMetricName('rmse').evaluate(predictions_trainData)
print(f"RMSE is {RMSE_trainData}")

RMSE is 0.223689933412301594
```



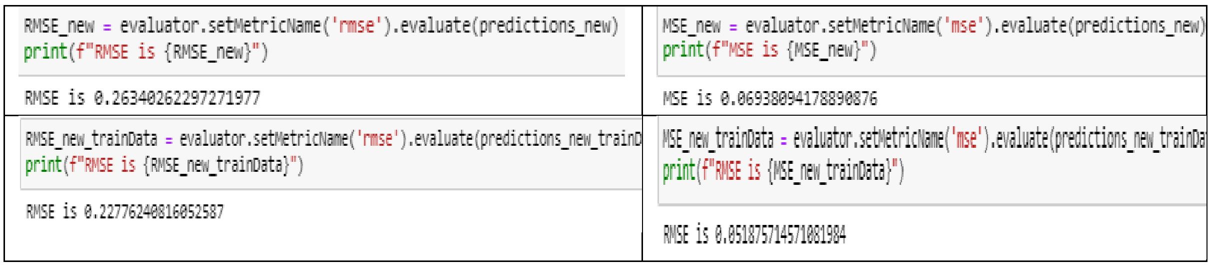
```
MSE_trainData = evaluator.setMetricName('mse').evaluate(predictions_trainData)
print(f"MSE is {MSE_trainData}")

MSE is 0.05003718662795921
```

There were ten features used in prediction, while feature "Car" contributed less than 1% to outcome prediction. Excluding this feature in modelling could reduce computing resources.



	feature	importance
9	SellerG_Indexed	0.310629
0	Rooms	0.226974
1	Distance	0.125013
6	Lattitude	0.093284
7	Longtitude	0.074430
2	Bedroom2	0.069207
5	Landsize	0.046613
3	Bathroom	0.026246
8	Propertycount	0.019504
4	Car	0.008101

The optimised RFR model performed slightly worse than the previous model on both metrics, which is reasonable.



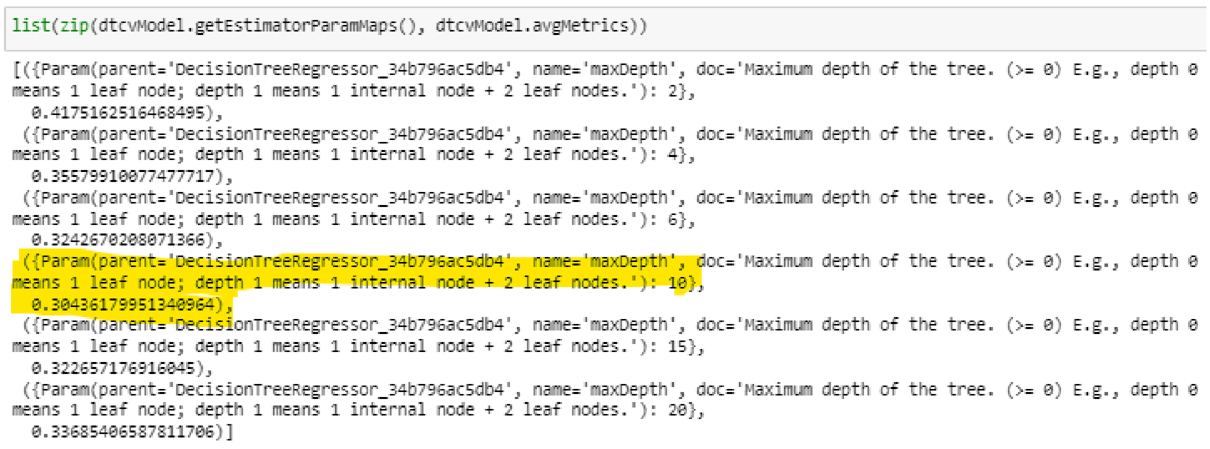
```
RMSE_new = evaluator.setMetricName('rmse').evaluate(predictions_new)
print(f"RMSE is {RMSE_new}")

RMSE is 0.26340262297271977
```

```
MSE_new = evaluator.setMetricName('mse').evaluate(predictions_new)
print(f"MSE is {MSE_new}")

MSE is 0.06938094178890876
```

```
RMSE_new_trainData = evaluator.setMetricName('rmse').evaluate(predictions_new_trainD
print(f"RMSE is {RMSE_new_trainData}")

RMSE is 0.22776240816052587
```

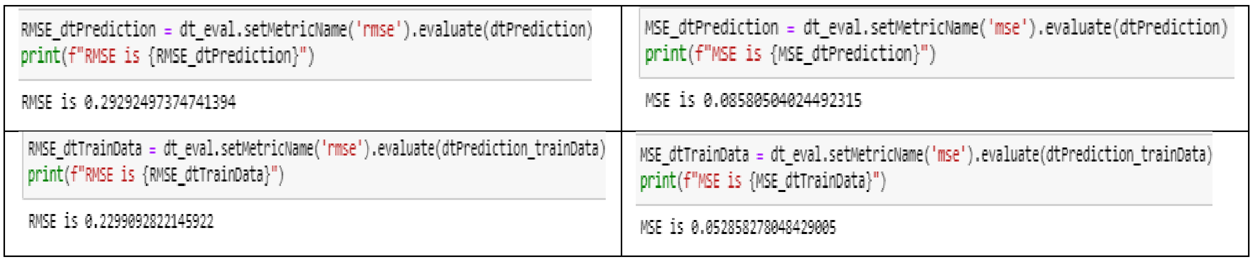
```
MSE_new_trainData = evaluator.setMetricName('mse').evaluate(predictions_new_trainDa
print(f"RMSE is {MSE_new_trainData}")

RMSE is 0.051875714571081984
```

Next, this project explored the DTR model performance. Hyperparameter tuning results showed the maximum tree depth of 10 based on metric rmse, where the decision tree could better learn the data pattern instead of memorising the data.



```
list(zip(dtcvModel.getEstimatorParamMaps(), dtcvModel.avgMetrics))

[({Param(parent='DecisionTreeRegressor_34b796ac5db4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 2},
  0.4175162516468495),
 ({Param(parent='DecisionTreeRegressor_34b796ac5db4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 4},
  0.35579910077477717),
 ({Param(parent='DecisionTreeRegressor_34b796ac5db4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 6},
  0.3242670208071366),
 ({Param(parent='DecisionTreeRegressor_34b796ac5db4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 10},
  0.30436179951340964),
 ({Param(parent='DecisionTreeRegressor_34b796ac5db4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 15},
  0.322657176916045),
 ({Param(parent='DecisionTreeRegressor_34b796ac5db4', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0
means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.'): 20},
  0.33685406587811706)]
```

The variances between the testing dataset and training dataset were a bit higher than the RFR model on both metrics, while variances were acceptable.



```
RMSE_dtPrediction = dt_eval.setMetricName('rmse').evaluate(dtPrediction)
print(f"RMSE is {RMSE_dtPrediction}")

RMSE is 0.29292497374741394
```

```
MSE_dtPrediction = dt_eval.setMetricName('mse').evaluate(dtPrediction)
print(f"MSE is {MSE_dtPrediction}")

MSE is 0.08580504024492315
```

```
RMSE_dtTrainData = dt_eval.setMetricName('rmse').evaluate(dtPrediction_trainData)
print(f"RMSE is {RMSE_dtTrainData}")

RMSE is 0.2299092822145922
```

```
MSE_dtTrainData = dt_eval.setMetricName('mse').evaluate(dtPrediction_trainData)
print(f"MSE is {MSE_dtTrainData}")

MSE is 0.052858278048429005
```

Summary of the discussed models as shown below,

Random Forest initial model						Random Forest optimised model						Decision Tree model with optimised features				
Hyperparameters		Test dataset		Train dataset		Hyperparameters		Test dataset		Train dataset		Hyperparameters	Test dataset		Train dataset	
maxDepth	numTrees	RMSE	MSE	RMSE	MSE	maxDepth	numTrees	RMSE	MSE	RMSE	MSE	maxDepth	RMSE	MSE	RMSE	MSE
10	100	0.26	0.068	0.224	0.05	10	100	0.263	0.069	0.228	0.052	10	0.293	0.086	0.23	0.053

The random forest model outperformed the decision tree model in house price prediction when examining in both metrics “rmse” and “mse”. Additionally, the performance of models outperformed the existing study conducted by (Phan, 2018) when comparing the tree-based model based on the outcome variable of logarithmic price. Phan (2018) reported the MSE value of 0.0925 while the results listed in the current project were 0.068, 0.069, and 0.086, as shown above. This could suggest that distributed computing probably did not impact model performance while it allowed machine learning to be scaled up with big data.

There are challenges during project execution. Firstly, data processing could closely relate to model selection. During the initial phase, linear regression was first considered model. However, the data visualisation showed poor linearity between some independent variables and the outcome variable. This project decided to choose tree-based models which are robust to extremes values, correlations among variables or lack of linearity. Secondly, exploring hyperparameters via cross-validation took a considerably long time. This could be an issue when the dataset is scaled up. It is worth effort on cross-validation model optimising to save computing power. Some processes, such as string indexing, generated constant output when running a cross-validation pipeline every time. Hence, this project demonstrated pipeline model optimisation by putting cross-validation inside the pipeline instead of putting the pipeline inside the cross-validation model.

However, this project did not explore many other machine learning models such as linear regression models due to the project focus being predicting the house prices by distributed computing system instead of exploring statistics assumptions. Additionally, further exploration of statistics could exceed the project’s required word count. Also, this project did not compare the computing efficiency of between optimised model and the original model, which is important when examining distributed computing system efficiency. Again, this is constrained by the scope and word count of the project. Additionally, the project demonstrated how to optimise the cross-validation process without comparing the difference in computing efficiencies. Future studies could be made on exploring these aspects.

References

Ho, W. K.O., Tang, B. and Wong, S.W. (2021) 'Predicting property prices with machine learning algorithms', Journal of Property Research, 38(1) [Online]. Available at: https://doi.org/10.1080/09599916.2020.1832558 (Accessed: 20 Mar 2022)

Drabas, T. and Lee, D. (2017) Learning PySpark. 1st edn. Birmingham: Packt Publishing Ltd

Gutierrez, D. (2015) Apache Spark is the Smartphone of Big Data [Online]. Available at: https://insidebigdata.com/2015/11/09/apache-spark-is-the-smartphone-of-big-data/ (Accessed: 20 Mar 2022)

Johnson, J. (2021) Number of emails per day worldwide 2017-2025 [Online]. Available at: https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/ (Accessed: 20 Mar 2022

Phan, T., D. 'Housing Price Prediction Using Machine Learning Algorithms: The Case of Melbourne City, Australia', 2018 International Conference on Machine Learning and Data Engineering (iCMLDE), [Online]. Available at: https://ieeexplore.ieee.org/document/8614000 (Accessed: 20 Mar 2022)

Pino, T. (2018) Melbourne housing clearance data from Jan 2016 [Online]. Available at: https://www.kaggle.com/datasets/anthonypino/melbourne-housing-market (Accessed: 20 Mar 2022)