

project3

October 29, 2021

1 AMS 586: Project-3

In this project we use pyspark to predict auction sale price for a piece of heavy equipment based on its usage, equipment type, and configuration.

1.1 Exploratory data analysis

In the following command we start pyspark and load the data. There are 300000 observations in the train data, 101125 observations in the test data and 53 variables. The predictor variable is 'SalePrice'.

```
[2]: import pyspark
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName('project3').getOrCreate()
train=spark.read.csv("D:\Python_D_Drive\SBU\AMS598\project3\Train_project3.
↪csv", header=True, inferSchema=True)
test=spark.read.csv("D:\Python_D_Drive\SBU\AMS598\project3\Test_project3.csv",
↪header=True, inferSchema=True)
```

```
[10]: train.select('SalesID').count(),len(train.columns)
```

```
[10]: (300000, 53)
```

```
[11]: test.select('SalesID').count(),len(test.columns)
```

```
[11]: (101125, 52)
```

The 'saledate' is converted to day, month and year. For this we load the necessary functions. Then we change each of those columns into IntegerType. Datatypes can be found using the print.Schema() command.

```
[5]: from pyspark.sql.functions import split,col
```

```
[6]: t = train.withColumn('date', split(train['saledate'], ' ').getItem(0)).
↪withColumn('salesmonth', split(train['saledate'], '/').getItem(0)).
↪withColumn('salesdate', split(train['saledate'], '/').getItem(1))
t1 = t.withColumn('salesyear', split(t['date'], '/').getItem(2))
```

```
[7]: from pyspark.sql.types import IntegerType
```

```
[8]: t2=t1.drop('date').drop('saledate').withColumn("salesyear",col("salesyear").
      ↳cast(IntegerType()).withColumn("salesdate",col("salesdate").
      ↳cast(IntegerType()).withColumn("salesmonth",col("salesmonth").
      ↳cast(IntegerType()))
```

```
[9]: t2.printSchema()
```

```
root
|-- SalesID: integer (nullable = true)
|-- SalePrice: integer (nullable = true)
|-- MachineID: integer (nullable = true)
|-- ModelID: integer (nullable = true)
|-- datasource: integer (nullable = true)
|-- auctioneerID: string (nullable = true)
|-- YearMade: integer (nullable = true)
|-- MachineHoursCurrentMeter: string (nullable = true)
|-- UsageBand: string (nullable = true)
|-- fiModelDesc: string (nullable = true)
|-- fiBaseModel: string (nullable = true)
|-- fiSecondaryDesc: string (nullable = true)
|-- fiModelSeries: string (nullable = true)
|-- fiModelDescriptor: string (nullable = true)
|-- ProductSize: string (nullable = true)
|-- fiProductClassDesc: string (nullable = true)
|-- state: string (nullable = true)
|-- ProductGroup: string (nullable = true)
|-- ProductGroupDesc: string (nullable = true)
|-- Drive_System: string (nullable = true)
|-- Enclosure: string (nullable = true)
|-- Forks: string (nullable = true)
|-- Pad_Type: string (nullable = true)
|-- Ride_Control: string (nullable = true)
|-- Stick: string (nullable = true)
|-- Transmission: string (nullable = true)
|-- Turbocharged: string (nullable = true)
|-- Blade_Extension: string (nullable = true)
|-- Blade_Width: string (nullable = true)
|-- Enclosure_Type: string (nullable = true)
|-- Engine_Horsepower: string (nullable = true)
|-- Hydraulics: string (nullable = true)
|-- Pushblock: string (nullable = true)
|-- Ripper: string (nullable = true)
|-- Scarifier: string (nullable = true)
|-- Tip_Control: string (nullable = true)
|-- Tire_Size: string (nullable = true)
|-- Coupler: string (nullable = true)
|-- Coupler_System: string (nullable = true)
```

```

|-- Grouser_Tracks: string (nullable = true)
|-- Hydraulics_Flow: string (nullable = true)
|-- Track_Type: string (nullable = true)
|-- Undercarriage_Pad_Width: string (nullable = true)
|-- Stick_Length: string (nullable = true)
|-- Thumb: string (nullable = true)
|-- Pattern_Changer: string (nullable = true)
|-- Grouser_Type: string (nullable = true)
|-- Backhoe_Mounting: string (nullable = true)
|-- Blade_Type: string (nullable = true)
|-- Travel_Controls: string (nullable = true)
|-- Differential_Type: string (nullable = true)
|-- Steering_Controls: string (nullable = true)
|-- salesmonth: integer (nullable = true)
|-- salesdate: integer (nullable = true)
|-- salesyear: integer (nullable = true)

```

1.2 Remove columns that does not have an influence on our model

Columns like 'SalesID','datasource','auctioneerID','state' don't explain anything about 'usage','equipment type','configuration' of the items. So we remove these columns.

```
[12]: t3 = t2.drop('state').drop('datasource').drop('auctioneerID').drop('SalesID')
```

1.3 Missing values and Handling categorical features

We now check the number of missing values in each of the columns and remove the columns that has less than 5% of non-null entries

```
[13]: # Find Count of Null, None, NaN of All DataFrame Columns
from pyspark.sql.functions import isnan, when, count
countNull = t3.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c
    ↪ c in t3.columns])
```

```
[14]: countNull.select(countNull.columns[0:9]).show()
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
---+-----+-----+-----+
|SalePrice|MachineID|ModelID|YearMade|MachineHoursCurrentMeter|UsageBand|fiModel
Desc|fiBaseModel|fiSecondaryDesc|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
---+-----+-----+-----+
|          0|          0|          0|          0|          0| 265273|
0|          0|      108774|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
---+-----+-----+-----+

```

```
[15]: countNull.select(countNull.columns[9:16]).show()
```

```
+-----+-----+-----+-----+-----+-----+
|fiModelSeries|fiModelDescriptor|ProductSize|fiProductClassDesc|ProductGroup|Pro
ductGroupDesc|Drive_System|
+-----+-----+-----+-----+-----+-----+
|      260544|      255078|    158665|          0|          0|
0|      85060|
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```
[16]: countNull.select(countNull.columns[16:26]).show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
|Enclosure|Forks|Pad_Type|Ride_Control|
Stick|Transmission|Turbocharged|Blade_Extension|Blade_Width|Enclosure_Type|
+-----+-----+-----+-----+-----+-----+-----+
|  135082|20239|  238385|    187137|238385|    220000|    179558|
281700|    281700|    281700|
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```
[17]: countNull.select(countNull.columns[26:36]).show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
|Engine_Horsepower|Hydraulics|Pushblock|Ripper|Scarifier|Tip_Control|Tire_Size|C
oupler|Coupler_System|Grouser_Tracks|
+-----+-----+-----+-----+-----+-----+-----+
|      281700|    197222|    147187|281629|    222799|    281700|    230456|
215950|    194059|    267997|
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```
[18]: countNull.select(countNull.columns[36:44]).show()
```

```
+-----+-----+-----+-----+-----+-----+
|Hydraulics_Flow|Track_Type|Undercarriage_Pad_Width|Stick_Length|
Thumb|Pattern_Changer|Grouser_Type|Backhoe_Mounting|
+-----+-----+-----+-----+-----+-----+
```

```

+-----+-----+-----+-----+-----+-----+
-----+-----+
|          267997|    300000|          226947|    226356|226891|
226854|          226891|          226947|
+-----+-----+-----+-----+
-----+-----+

```

```
[19]: countNull.select(countNull.columns[44:len(t3.columns)]).show()
```

```

+-----+-----+-----+-----+-----+-----+
----+-----+
|Blade_Type|Travel_Controls|Differential_Type|Steering_Controls|salesmonth|sales
date|salesyear|
+-----+-----+-----+-----+-----+-----+
----+-----+
|    241244|          240216|          189026|          248841|          0|
0|          0|
+-----+-----+-----+-----+-----+-----+
----+-----+

```

```
[20]: #Then number of null entries will be
300000-300000*.05
```

```
[20]: 285000.0
```

```
[21]: #Track_Type has no non-null entries at all
train1=train1.drop('Track_Type')
```

We now create a list of columns with string data types and integer datatypes, to fill the missing values. From the results we can see that integer entries have no missing values at all. To deal with the missing values of the string entries, we converted them to categorical values and imputed the missing values by 0.

```
[22]: #create a list of the columns that are string typed
categoricalColumns = [item[0] for item in train1.dtypes if item[1].
↳startswith('string')]
#create a list of the columns that are int typed
numericColumns = [item[0] for item in train1.dtypes if item[1].
↳startswith('int')]

```

Correlation matrix of integer entries

```
[23]: numeric_data = train1.select(numericColumns).toPandas()
numeric_data.corr()
```

```
[23]:
```

	SalePrice	MachineID	ModelID	YearMade	salesmonth	salesdate	\
SalePrice	1.000000	-0.254037	-0.083643	0.159629	-0.037443	0.000472	
MachineID	-0.254037	1.000000	0.095026	-0.070771	-0.006996	-0.005424	
ModelID	-0.083643	0.095026	1.000000	-0.064605	0.001178	0.002503	
YearMade	0.159629	-0.070771	-0.064605	1.000000	0.010332	-0.008293	
salesmonth	-0.037443	-0.006996	0.001178	0.010332	1.000000	-0.116913	
salesdate	0.000472	-0.005424	0.002503	-0.008293	-0.116913	1.000000	
salesyear	0.012833	-0.108571	0.109355	-0.057143	-0.052192	0.027176	

	salesyear
SalePrice	0.012833
MachineID	-0.108571
ModelID	0.109355
YearMade	-0.057143
salesmonth	-0.052192
salesdate	0.027176
salesyear	1.000000

```
[24]: import pandas as pd
```

```
[25]: data=train1.toPandas()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300000 entries, 0 to 299999
Data columns (total 50 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   SalePrice                             300000 non-null  int32
1   MachineID                             300000 non-null  int32
2   ModelID                               300000 non-null  int32
3   YearMade                              300000 non-null  int32
4   MachineHoursCurrentMeter              300000 non-null  object
5   UsageBand                             34727 non-null   object
6   fiModelDesc                           300000 non-null  object
7   fiBaseModel                           300000 non-null  object
8   fiSecondaryDesc                       191226 non-null  object
9   fiModelSeries                         39456 non-null   object
10  fiModelDescriptor                     44922 non-null   object
11  ProductSize                           141335 non-null  object
12  fiProductClassDesc                    300000 non-null  object
13  ProductGroup                          300000 non-null  object
14  ProductGroupDesc                      300000 non-null  object
15  Drive_System                          214940 non-null  object
16  Enclosure                             164918 non-null  object
17  Forks                                 279761 non-null  object
18  Pad_Type                              61615 non-null   object
```

```

19 Ride_Control          112863 non-null object
20 Stick                 61615 non-null object
21 Transmission          80000 non-null object
22 Turbocharged          120442 non-null object
23 Blade_Extension       18300 non-null object
24 Blade_Width           18300 non-null object
25 Enclosure_Type        18300 non-null object
26 Engine_Horsepower     18300 non-null object
27 Hydraulics            102778 non-null object
28 Pushblock             152813 non-null object
29 Ripper                 18371 non-null object
30 Scarifier              77201 non-null object
31 Tip_Control           18300 non-null object
32 Tire_Size             69544 non-null object
33 Coupler                84050 non-null object
34 Coupler_System        105941 non-null object
35 Grouser_Tracks         32003 non-null object
36 Hydraulics_Flow       32003 non-null object
37 Undercarriage_Pad_Width 73053 non-null object
38 Stick_Length          73644 non-null object
39 Thumb                 73109 non-null object
40 Pattern_Changer       73146 non-null object
41 Grouser_Type           73109 non-null object
42 Backhoe_Mounting      73053 non-null object
43 Blade_Type            58756 non-null object
44 Travel_Controls       59784 non-null object
45 Differential_Type     110974 non-null object
46 Steering_Controls     51159 non-null object
47 salesmonth            300000 non-null int32
48 salesdate             300000 non-null int32
49 salesyear             300000 non-null int32
dtypes: int32(7), object(43)
memory usage: 106.4+ MB

```

```

[26]: for label,content in data.items():

        #in the new dataset , search for string data type cintents and convert them
        ↳to category type

        if pd.api.types.is_string_dtype(content):
            data[label] = content.astype("category").cat.as_ordered()

```

```

[27]: for label,value in data.items():
        if not pd.api.types.is_numeric_dtype(value):

            data[label] = pd.Categorical(value).codes+1

```

```
[28]: data.to_csv("train_modified.csv")
```

```
[29]: train_modified=spark.read.csv("train_modified.csv", header=True,
    ↪inferSchema=True)
```

```
[30]: train_modified1=train_modified.drop('_c0')
```

1.4 Linear Regression

We index the categorical variables using the StringIndexer. For regression, we first form a n-tuple of the features and call them as IndependentFeature using Vector Assembler. Using this IndependentFeature, we first divide the data into train and valid data, and then with this model we fit the test_data to predict the SalePrice.

```
[31]: from pyspark.ml.feature import StringIndexer
```

```
[32]: Indexer = StringIndexer(inputCols = categoricalColumns, outputCols =
    ↪['MachineHoursCurrentMeter_index',
    'UsageBand_index',
    'fiModelDesc_index',
    'fiBaseModel_index',
    'fiSecondaryDesc_index',
    'fiModelSeries_index',
    'fiModelDescriptor_index',
    'ProductSize_index',
    'fiProductClassDesc_index',
    'ProductGroup_index',
    'ProductGroupDesc_index',
    'Drive_System_index',
    'Enclosure_index',
    'Forks_index',
    'Pad_Type_index',
    'Ride_Control_index',
    'Stick_index',
    'Transmission_index',
    'Turbocharged_index',
    'Blade_Extension_index',
    'Blade_Width_index',
    'Enclosure_Type_index',
    'Engine_Horsepower_index',
    'Hydraulics_index',
    'Pushblock_index',
    'Ripper_index',
    'Scarifier_index',
    'Tip_Control_index',
    'Tire_Size_index',
    'Coupler_index',
```



```

'Coupler_System_index',
'Grouser_Tracks_index',
'Hydraulics_Flow_index',
'Undercarriage_Pad_Width_index',
'Stick_Length_index',
'Thumb_index',
'Pattern_Changer_index',
'Grouser_Type_index',
'Backhoe_Mounting_index',
'Blade_Type_index',
'Travel_Controls_index',
'Differential_Type_index',
'Steering_Controls_index'])
train_modified2=Indexer.fit(train_modified1).transform(train_modified1)

```

```

[34]: #Non-index columns are removed
train_modified3=train_modified2.drop('MachineHoursCurrentMeter').drop(
'UsageBand').drop(
'fiModelDesc').drop(
'fiBaseModel').drop(
'fiSecondaryDesc').drop(
'fiModelSeries').drop(
'fiModelDescriptor').drop(
'ProductSize').drop(
'fiProductClassDesc').drop(
'ProductGroup').drop(
'ProductGroupDesc').drop(
'Drive_System').drop(
'Enclosure').drop(
'Forks').drop(
'Pad_Type').drop(
'Ride_Control').drop(
'Stick').drop(
'Transmission').drop(
'Turbocharged').drop(
'Blade_Extension').drop(
'Blade_Width').drop(
'Enclosure_Type').drop(
'Engine_Horsepower').drop(
'Hydraulics').drop(
'Pushblock').drop(
'Ripper').drop(
'Scarifier').drop(
'Tip_Control').drop(
'Tire_Size').drop(
'Coupler').drop(
'Coupler_System').drop(

```

```

'Grouser_Tracks').drop(
'Hydraulics_Flow').drop(
'Undercarriage_Pad_Width').drop(
'Stick_Length').drop(
'Thumb').drop(
'Pattern_Changer').drop(
'Grouser_Type').drop(
'Backhoe_Mounting').drop(
'Blade_Type').drop(
'Travel_Controls').drop(
'Differential_Type').drop(
'Steering_Controls')
train_modified3.toPandas().head(5)

```

```

[34]:   SalePrice  MachineID  ModelID  YearMade  salesmonth  salesdate  salesyear  \
0      66000     999089     3157     2004           11          16         2006
1      57000     117657         77     1996           3          26         2004
2      10000     434808     7009     2001           2          26         2004
3      38500    1026470        332     2001           5          19         2011
4      11000    1057373    17311     2007           7          23         2009

```

```

    Tip_Control_index  Steering_Controls_index  Drive_System_index  ...  \
0                0.0                1.0                0.0  ...
1                0.0                1.0                0.0  ...
2                0.0                0.0                0.0  ...
3                0.0                0.0                1.0  ...
4                0.0                0.0                0.0  ...

```

```

    ProductGroup_index  Tire_Size_index  Ride_Control_index  \
0                1.0                1.0                2.0
1                1.0                4.0                2.0
2                2.0                0.0                0.0
3                5.0                0.0                0.0
4                2.0                0.0                0.0

```

```

    Grouser_Type_index  UsageBand_index  Differential_Type_index  \
0                0.0                2.0                2.0
1                0.0                2.0                2.0
2                0.0                3.0                0.0
3                0.0                3.0                0.0
4                0.0                1.0                0.0

```

```

    fiModelSeries_index  Blade_Extension_index  ProductGroupDesc_index  \
0                0.0                0.0                3.0
1                1.0                0.0                3.0
2                0.0                0.0                4.0
3                53.0                0.0                0.0

```

4	0.0	0.0	4.0
---	-----	-----	-----

	fiBaseModel_index
0	429.0
1	5.0
2	78.0
3	86.0
4	96.0

[5 rows x 50 columns]

```
[35]: from pyspark.ml.feature import VectorAssembler
```

```
[36]: featureassembler=VectorAssembler(inputCols=['MachineID',
'ModelID',
'YearMade',
'salesyear',
'Tip_Control_index',
'Steering_Controls_index',
'Drive_System_index',
'Grouser_Tracks_index',
'MachineHoursCurrentMeter_index',
'Enclosure_index',
'Engine_Horsepower_index',
'fiModelDesc_index',
'Blade_Width_index',
'Stick_Length_index',
'Ripper_index',
'fiProductClassDesc_index',
'fiModelDescriptor_index',
'Coupler_System_index',
'Stick_index',
'fiSecondaryDesc_index',
'Undercarriage_Pad_Width_index',
'Enclosure_Type_index',
'Pushblock_index',
'Scarifier_index',
'Hydraulics_Flow_index',
'Pattern_Changer_index',
'Forks_index',
'Blade_Type_index',
'Backhoe_Mounting_index',
'ProductSize_index',
'Coupler_index',
'Pad_Type_index',
'Transmission_index',
'Travel_Controls_index',
```

```
'Hydraulics_index',
'Thumb_index',
'Turbocharged_index',
'ProductGroup_index',
'Tire_Size_index',
'Ride_Control_index',
'Grouser_Type_index',
'UsageBand_index',
'Differential_Type_index',
'fiModelSeries_index',
'Blade_Extension_index',
'ProductGroupDesc_index',
'fiBaseModel_index'], outputCol="IndependentFeatures")
```

```
[37]: output=featureassembler.transform(train_modified3)
```

```
[38]: finalized_data=output.select(['IndependentFeatures','SalePrice'])
finalized_data.show()
```

```
+-----+-----+
| IndependentFeatures|SalePrice|
+-----+-----+
|(47,[0,1,2,3,5,8,...|    66000|
|(47,[0,1,2,3,5,8,...|    57000|
|(47,[0,1,2,3,7,8,...|    10000|
|(47,[0,1,2,3,6,8,...|    38500|
|(47,[0,1,2,3,7,8,...|    11000|
|(47,[0,1,2,3,6,8,...|    26500|
|(47,[0,1,2,3,6,8,...|    21000|
|(47,[0,1,2,3,6,8,...|    27000|
|(47,[0,1,2,3,6,8,...|    21500|
|(47,[0,1,2,3,5,8,...|    65000|
|(47,[0,1,2,3,6,8,...|    24000|
|(47,[0,1,2,3,6,8,...|    22500|
|(47,[0,1,2,3,6,8,...|    36000|
|(47,[0,1,2,3,6,8,...|    30500|
|(47,[0,1,2,3,6,8,...|    28000|
|(47,[0,1,2,3,6,8,...|    19000|
|(47,[0,1,2,3,6,8,...|    13500|
|(47,[0,1,2,3,6,8,...|     9500|
|(47,[0,1,2,3,6,8,...|    12500|
|(47,[0,1,2,3,6,8,...|    11500|
+-----+-----+
only showing top 20 rows
```

```
[39]: train_data,valid_data=finalized_data.randomSplit([0.75,0.25])
```

```
[40]: from pyspark.ml.regression import LinearRegression
```

```
[41]: regressor = LinearRegression(featuresCol='IndependentFeatures',  
    ↪labelCol='SalePrice')
```

```
[42]: regressor=regressor.fit(train_data)
```

```
[43]: regressor.coefficients
```

```
[43]: DenseVector([-0.0087, 0.0024, 10.2399, -103.7967, 1168.6724, 5294.139,  
-2331.7174, 644.4963, -0.1765, 7211.8932, 19299.4972, 0.4324, -2864.7395,  
-77.5617, 16379.1353, 57.2345, 384.654, 1253.0401, -1803.4434, 303.1317,  
-16069.1523, 15118.5802, 80.4105, 4759.414, 11672.7314, 1653.08, 2375.9532,  
11906.8134, 4762.6989, -646.5276, 312.6049, -2080.7804, -2064.8926, 2745.3525,  
-442.5539, 1026.5157, -1162.0654, -112.8343, 641.9305, 4289.8443, 12413.2203,  
2154.3741, 2057.8117, 84.9877, -6280.417, -10121.9464, -10.8379])
```

```
[44]: regressor.intercept
```

```
[44]: 229110.16452802604
```

```
[45]: pred_results = regressor.evaluate(valid_data)
```

```
[46]: pred_results.predictions.show()
```

```
D:\Anaconda\lib\site-packages\pyspark\sql\context.py:125: FutureWarning:  
Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.  
    warnings.warn(  

```

```
+-----+-----+-----+  
| IndependentFeatures|SalePrice|      prediction|  
+-----+-----+-----+  
|(47,[0,1,2,3,4,6,...|    44000| 51860.10494487657|  
|(47,[0,1,2,3,4,6,...|    47000|47070.915824142954|  
|(47,[0,1,2,3,4,6,...|    55000| 54118.24127163185|  
|(47,[0,1,2,3,4,6,...|    89000| 49620.12165392228|  
|(47,[0,1,2,3,4,6,...|    27000| 67148.00506173004|  
|(47,[0,1,2,3,4,6,...|    29500| 48969.36027674665|  
|(47,[0,1,2,3,4,6,...|    56000| 52491.54701729183|  
|(47,[0,1,2,3,4,6,...|    50000| 52744.91890354236|  
|(47,[0,1,2,3,4,6,...|    70000| 59701.78844269435|  
|(47,[0,1,2,3,4,6,...|    84000| 48519.93005413411|  
|(47,[0,1,2,3,4,6,...|    52000| 46034.27187943173|  
|(47,[0,1,2,3,4,6,...|    58000| 72157.17127247236|  
|(47,[0,1,2,3,4,6,...|    48000| 48478.17316041197|  
|(47,[0,1,2,3,4,6,...|    42000| 43982.46504434961|  
|(47,[0,1,2,3,4,6,...|    33000| 41078.04499500923|  
|(47,[0,1,2,3,4,6,...|    69000| 67228.28756350989|
```

```
| (47, [0,1,2,3,4,6,...|      30000|55606.752533259685|
| (47, [0,1,2,3,4,6,...|      39000| 58422.99193862427|
| (47, [0,1,2,3,4,6,...|      46000| 48596.09004754454|
| (47, [0,1,2,3,4,6,...|      31000|31398.277866307704|
+-----+-----+-----+
only showing top 20 rows
```

```
[47]: pred_results.meanAbsoluteError, pred_results.meanSquaredError
```

```
[47]: (12638.082867944751, 306911148.61585015)
```

2 Prediction of test data

We perform the necessary pre-processing to the test data and predict the SalePrice using the model mentioned above

```
[48]: test1 = test.withColumn('date', split(test['saledate'], ' ').getItem(0)).
      ↳withColumn('salesmonth', split(test['saledate'], '/').getItem(0)).
      ↳withColumn('salesdate', split(test['saledate'], '/').getItem(1))
test2 = test1.withColumn('salesyear', split(test1['date'], '/').getItem(2))
test3=test2.drop('date').drop('saledate').
      ↳withColumn("salesyear",col("salesyear").cast(IntegerType())).
      ↳withColumn("salesdate",col("salesdate").cast(IntegerType())).
      ↳withColumn("salesmonth",col("salesmonth").cast(IntegerType()))
test4 = test3.drop('state').drop('datasource').drop('auctioneerID').
      ↳drop('SalesID').drop('Track_Type')
```

```
[49]: test4.toPandas().head(5)
```

```
[49]:
```

	MachineID	ModelID	YearMade	MachineHoursCurrentMeter	UsageBand	\
0	285987	13776	2000		0	None
1	115917	13776	2001		0	None
2	1714094	13776	1999		0	None
3	566454	13776	1000		2737	Medium
4	650474	13776	2001		5665	High

	fiModelDesc	fiBaseModel	fiSecondaryDesc	fiModelSeries	fiModelDescriptor	\
0	D3CIIIXL	D3	C	III		XL
1	D3CIIIXL	D3	C	III		XL
2	D3CIIIXL	D3	C	III		XL
3	D3CIIIXL	D3	C	III		XL
4	D3CIIIXL	D3	C	III		XL

	...	Pattern_Changer	Grouser_Type	Backhoe_Mounting	Blade_Type	\
0	...	None	None	None	None or Unspecified	
1	...	None	None	None	None or Unspecified	

2	...	None	None	None	None or Unspecified
3	...	None	None	None	None or Unspecified
4	...	None	None	None	None or Unspecified

	Travel_Controls	Differential_Type	Steering_Controls	salesmonth	\
0	None or Unspecified	None or Unspecified	None	5	
1	PAT	None or Unspecified	None	6	
2	PAT	None or Unspecified	None	7	
3	PAT	None or Unspecified	None	5	
4	PAT	None or Unspecified	None	5	

	salesdate	salesyear
0	11	2007
1	30	2010
2	14	2007
3	13	2010
4	5	2010

[5 rows x 49 columns]

```
[50]: test_data=test4.toPandas()
```

```
[51]: for label,content in test_data.items():

        #in the new dataset , search for string data type cintents and convert them
        ↪to category type

        if pd.api.types.is_string_dtype(content):
            test_data[label] = content.astype("category").cat.as_ordered()
```

```
[52]: for label,value in test_data.items():
        if not pd.api.types.is_numeric_dtype(value):

            test_data[label] = pd.Categorical(value).codes+1
```

```
[53]: test_data.to_csv("test_modified.csv")
```

```
[54]: test_modified=spark.read.csv("test_modified.csv", header=True, inferSchema=True)
```

```
[55]: test_modified=test_modified.drop('_c0')
```

```
[56]: test_modified=Indexer.fit(test_modified).transform(test_modified)
```

```
[57]: test_modified=test_modified.drop('MachineHoursCurrentMeter').drop(
        'UsageBand').drop(
        'fiModelDesc').drop(
        'fiBaseModel').drop(
```

```

'fiSecondaryDesc').drop(
'fiModelSeries').drop(
'fiModelDescriptor').drop(
'ProductSize').drop(
'fiProductClassDesc').drop(
'ProductGroup').drop(
'ProductGroupDesc').drop(
'Drive_System').drop(
'Enclosure').drop(
'Forks').drop(
'Pad_Type').drop(
'Ride_Control').drop(
'Stick').drop(
'Transmission').drop(
'Turbocharged').drop(
'Blade_Extension').drop(
'Blade_Width').drop(
'Enclosure_Type').drop(
'Engine_Horsepower').drop(
'Hydraulics').drop(
'Pushblock').drop(
'Ripper').drop(
'Scarifier').drop(
'Tip_Control').drop(
'Tire_Size').drop(
'Coupler').drop(
'Coupler_System').drop(
'Grouser_Tracks').drop(
'Hydraulics_Flow').drop(
'Undercarriage_Pad_Width').drop(
'Stick_Length').drop(
'Thumb').drop(
'Pattern_Changer').drop(
'Grouser_Type').drop(
'Backhoe_Mounting').drop(
'Blade_Type').drop(
'Travel_Controls').drop(
'Differential_Type').drop(
'Steering_Controls')

```

```
[58]: output_test=featureassembler.transform(test_modified)
```

```
[59]: finalized_test=output_test.select('IndependentFeatures')
finalized_test.show()
```

```

+-----+
| IndependentFeatures|

```



```

+-----+
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,8,...|
| (47, [0,1,2,3,6,8,...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,6,11...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
| (47, [0,1,2,3,5,9,...|
+-----+
only showing top 20 rows

```

```
[60]: pred_results = regressor.transform(finalized_test)
```

```
[61]: pred_results.select('prediction').show()
```

```

+-----+
|      prediction|
+-----+
| 53518.54991904719|
|56111.868501558114|
|41138.823656139546|
| 39802.39895797687|
| 62455.57186450189|
| 43922.49291816613|
| 44974.67969114549|
| 44319.17545328732|
|50085.191998659866|
| 49522.90517279276|
| 36008.21766934925|
| 46649.69387331695|
| 42455.46420502878|
| 42322.43769438728|
| 51490.57403848882|
|43232.811451850226|

```

```
| 40687.24149285717|
| 39975.97043458151|
| 51489.90756689245|
|51431.480087769276|
+-----+
only showing top 20 rows
```

```
[62]: result=pred_results.select('prediction').toPandas()
```

```
[63]: result.head(5)
```

```
[63]:      prediction
0  53518.549919
1  56111.868502
2  41138.823656
3  39802.398958
4  62455.571865
```

```
[64]: test_csv=pd.read_csv("D:\Python_D_Drive\SBU\AMS598\project3\Test_project3.csv",
    ↳ usecols=range(0,52),parse_dates=["saledate"],)
```

```
[65]: test_results = pd.concat([test_csv, result], axis=1)
```

```
[66]: test_results.head(5)
```

```
[66]:      SalesID  MachineID  ModelID  datasource  auctioneerID  YearMade  \
0  2241104      285987    13776         136           NaN      2000
1  2241111      115917    13776         136           1.0      2001
2  2241112     1714094    13776         136           NaN      1999
3  2241114      566454    13776         136           1.0      1000
4  2241118      650474    13776         136           1.0      2001

      MachineHoursCurrentMeter  UsageBand  saledate  fiModelDesc  ...  \
0                          0.0        NaN  2007-05-11    D3CIIIXL  ...
1                          0.0        NaN  2010-06-30    D3CIIIXL  ...
2                          0.0        NaN  2007-07-14    D3CIIIXL  ...
3                      2737.0    Medium  2010-05-13    D3CIIIXL  ...
4                      5665.0     High  2010-05-05    D3CIIIXL  ...

      Stick_Length  Thumb  Pattern_Changer  Grouser_Type  Backhoe_Mounting  \
0             NaN    NaN              NaN          NaN              NaN
1             NaN    NaN              NaN          NaN              NaN
2             NaN    NaN              NaN          NaN              NaN
3             NaN    NaN              NaN          NaN              NaN
4             NaN    NaN              NaN          NaN              NaN
```

	Blade_Type	Travel_Controls	Differential_Type	\
0	None or Unspecified	None or Unspecified	None or Unspecified	
1	None or Unspecified	PAT	None or Unspecified	
2	None or Unspecified	PAT	None or Unspecified	
3	None or Unspecified	PAT	None or Unspecified	
4	None or Unspecified	PAT	None or Unspecified	

	Steering_Controls	prediction
0	NaN	53518.549919
1	NaN	56111.868502
2	NaN	41138.823656
3	NaN	39802.398958
4	NaN	62455.571865

[5 rows x 53 columns]

```
[67]: test_results.to_csv("test_results.csv")
```

```
[ ]:
```