**REPORT**

* Dataset name: Another-Dataset-on-used-Fiat-500
* This dataset has been created from a query done on an website specialized in used cars and contains 1538 rows
* Description of columns (Features):
* model: Fiat 500 comes in several 'flavours’:'pop', 'lounge', 'sport' (3 distinct values).
* engine\_power: number of Kw of the engine.
* ageindays: age of the car in number of days (from the time the dataset has been created).
* km: kilometers of the car.
* previous\_owners: number of previous owners.
* lat: latitude of the seller (the price of cars in Italy varies from North to South of the country).
* lon: longitude of the seller (the price of cars in Italy varies from North to South of the country).
* price: selling price (the target).
* Features: Total 7 features. (6 numerical and 1 nominal)
* Target: price (selling price) is the target for given dataset and it is numerical. The dataset has 222 distinct values for target.

Table

Description automatically generated with low confidence

* Number of instances: 1538
* The dataset information before one hot encoding is as follows:

Text

Description automatically generated

* We have one nominal feature – “model” with 3 distinct values ['lounge' 'pop' 'sport'].
* The dataset information after one hot encoding is as follows:

Graphical user interface, text, application

Description automatically generated with medium confidence

**The setting of the methods:**

* We have zero missing attributes for all features.
* For decision tree regressor: parameter is used min\_sample\_leaf. Min\_sample\_leaf has five values: [2,4,6,8,10].
* For K Nearest Neighbors Regression: n\_neighbours are used which has values of k from one to ten and one as Manhattan distance.
* For all models, in model selection cross validate function cv is set to 10 and scoring to neg\_root\_mean\_squared\_error
* Also, in learning curve function train\_sizes are set to range of [0.2, 0.4, 0.6, 0.8, 1], cv is set to 10, return\_times is set to True; scoring is set to neg\_root\_mean\_squared\_error, shuffle is set to True, and random\_state is set to 0 for methods.

**List of models:**

|  |  |
| --- | --- |
| **Sr. No.** | **Model Name** |
| 1 | DecisionTreeRegressor |
| 2 | KNearestNeighborsRegressor |
| 3 | LinearRegression |
| 4 | Support Vector Machine regressor |
| 5 | Bagged decision tree regressor |
| 6 | DummyRegressor |

**The rmse values obtained for each model with increasing amount of training data using 10-fold cross-validation:**

|  |  |
| --- | --- |
| **Model Name** | **RMSE** |
| DecisionTreeRegressor | [[ 794.74121396 859.67219905 767.95294554 815.43428131 935.55365557  802.25046742 988.86497959 885.64236469 805.68222706 1049.38437395]  [ 759.92792134 866.58045969 718.31432933 837.92920658 943.97761786  707.36462437 957.24484617 897.1476168 765.51166799 931.95296693]  [ 712.29980364 844.32557477 712.31099878 862.65374343 940.23138519  738.52137476 918.08532059 814.06981679 747.847485 944.19928001]  [ 735.15654313 870.42406324 763.18362111 854.78079885 913.82409683  798.38113765 979.58958967 840.42385783 745.17894819 905.81282447]  [ 754.01963001 883.99099063 674.31608925 852.24720013 882.41961799  807.26990557 953.15329643 928.07079326 743.74131505 889.36287933]] |
| KNearestNeighborsRegressor | [[ 914.04645336 992.01375406 847.49992871 1091.77625987 1076.9203769  973.47688538 1085.18715577 830.68071202 889.59766488 1055.44063768]  [ 836.85154675 974.5137007 821.68469804 1061.82348351 1025.20173286  918.96891837 1020.32830055 848.70468798 829.266594 1029.4526929 ]  [ 803.73664623 949.99183749 803.3261752 978.21940186 1010.10491001  878.7664981 1031.3688392 875.87913149 829.4717019 958.01129518]  [ 783.68148024 920.84733784 775.91616029 933.84993973 985.83827175  887.05922598 995.77084942 851.28872965 819.32828865 899.56466141]  [ 774.17187091 895.66863466 771.93481169 919.83424502 959.0326847  882.85922077 954.200159 833.74794406 780.36339373 909.57715253]] |
| LinearRegression | [[655.90749209 766.35828505 718.84014561 766.97231394 794.14515012  701.43245999 911.10038027 820.98295364 677.89083389 889.61905294]  [629.3547252 767.38486089 717.4907004 765.04365264 793.918965  708.92792413 895.78691687 822.36542943 669.73933171 882.02123657]  [633.08797476 772.08987934 702.87015634 764.80377114 804.03991998  707.26774978 897.31402888 815.67798455 670.34900624 884.6921776 ]  [629.03484202 774.75334873 704.55535629 766.6354119 800.43399493  703.95681858 899.77806188 815.8384936 668.11660772 879.45006923]  [626.98886729 769.0222705 697.36520275 767.7060443 790.10754395  702.90289294 903.69359693 819.09197663 660.7360421 873.23340221]] |
| Support Vector Machine regressor | [[2141.05590619 2290.55166569 1823.913033 1854.65379309 2078.15666755  1908.59800674 1955.10199329 1764.09935856 1761.51546512 2047.72709332]  [2086.72035868 2222.02355893 1783.39966808 1834.79086593 2047.63236307  1883.47352708 1908.75534461 1727.92192728 1737.26403417 1885.8446106 ]  [2060.76090838 2163.87236737 1747.8717244 1826.30874156 2017.51982996  1879.52485626 1897.79743933 1694.24865749 1715.0848995 1858.03980408]  [2078.91356123 2094.04690972 1713.09022983 1790.89968478 1984.2253444  1838.24994049 1908.82636238 1661.78135742 1679.16195137 1834.32587848]  [2006.94390199 2034.43160621 1680.42202841 1752.02300698 1966.56202013  1830.45328199 1870.64715331 1627.14811297 1640.54433858 1833.65825424]] |
| Bagged decision tree regressor | [[ 755.09810771 803.87877717 724.32396954 866.97804392 917.93110792  743.52510233 986.94788547 897.90478612 748.8832059 1004.97731236]  [ 680.80947589 844.82721442 725.87022508 833.6974304 891.46215141  696.24988997 910.15740265 849.3245987 716.23591385 914.5885127 ]  [ 677.36694299 806.25842539 662.93642151 837.57552057 864.28144732  670.87764126 916.50344311 785.98685492 702.20845938 857.19252026]  [ 688.90420385 811.87341838 685.38182699 814.3866184 908.17197031  714.63484623 947.62370958 857.58160559 702.9608929 863.50511842]  [ 682.09535057 838.52329262 686.73130435 817.54643019 883.37831534  702.87620559 949.74282763 832.47005621 656.24306537 867.69024427]] |
| DummyRegressor | [[2046.48885292 2157.02763356 1887.16725234 1858.79561517 2102.46091683  1878.40161831 1945.30660094 1879.40051942 1792.04805011 1901.75955599]  [2045.5033506 2151.64605926 1890.87825028 1858.61421291 2092.4641988  1877.3512434 1945.41675244 1856.83115528 1788.90123797 1885.58621464]  [2049.1742838 2151.61163968 1889.1820955 1858.36661076 2092.32555242  1882.92363739 1946.14099878 1847.49441491 1786.25685073 1882.41496613]  [2057.806359 2149.57177328 1885.71597837 1858.38092352 2092.35949321  1876.90444637 1946.96766358 1839.95067984 1792.85416526 1880.15975584]  [2055.41050773 2148.35744373 1884.126465 1858.78031559 2092.34049452  1877.7788042 1946.40689799 1846.48721647 1789.98957133 1883.09787907]] |

**Let’s plot a graph showing the learning curves of all the above models:**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Model Name** | **Learning curve graph** |
| 1 | DecisionTreeRegressor | Chart, line chart  Description automatically generated |
| 2 | KNearestNeighborsRegressor | Chart, line chart  Description automatically generated |
| 3 | LinearRegression | Chart, line chart  Description automatically generated |
| 4 | Support Vector Machine regressor | Chart, line chart  Description automatically generated |
| 5 | Bagged decision tree regressor | Chart, line chart  Description automatically generated |
| 6 | DummyRegressor | Chart, line chart  Description automatically generated |

**The following table consists of rmse of all the models obtained using 10-fold cross-validation with full training data (i.e., last point of the learning curve)**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Model Name** | **RMSE (using last point of learning curve)** |
| 1 | DecisionTreeRegressor | 836.8591717654526 |
| 2 | KNearestNeighborsRegressor | 868.1390117084945 |
| **3** | **LinearRegression** | **761.0847839600915** |
| 4 | Support Vector Machine regressor | 1824.28337048039 |
| 5 | Bagged decision tree regressor | 791.7297092151385 |
| 6 | DummyRegressor | 1938.277559562325 |

* The LinearRegression model is the best performing model among all. Since rmse value is 761.08 and it is less than remaining models.

**Let’s compare LinearRegression model with each of the remaining models for statistical significance (Using two-tailed paired t-test from scipy):**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Model Name** | **Ttest\_relResult** |
| 1 | DecisionTreeRegressor\* | statistic=-11.204459173682976, pvalue=0.0003613015599417494 |
| 2 | KNearestNeighborsRegressor\* | statistic=-8.52647725747304, pvalue=0.0010381533423327433 |
| 3 | Support Vector Machine regressor\* | statistic=-50.84938827240463, pvalue=8.951367708806311e-07 |
| 4 | Bagged decision tree regressor\* | statistic=-4.298849091667956, pvalue=0.012657140654345386 |
| 5 | DummyRegressor\* | statistic=-1742.0877128940126, pvalue=6.514337060634445e-13 |

If pvalue > 0.05. It does not show they are statistically significantly different. On the other hand, if pvalue < 0.05, then they are statistically significantly different.

**Let’s compare the computational training and test times of all the models for the last point of the learning curve.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Model Name** | **Training times (fit\_times)** | **Testing times (score\_times)** |
| 1 | DecisionTreeRegressor | 0.24312202930450438 | 0.0008022308349609375 |
| 2 | KNearestNeighborsRegressor | 0.14352903366088868 | 0.0018187522888183593 |
| 3 | LinearRegression | 0.0028255224227905274 | 0.0011689901351928712 |
| 4 | Support Vector Machine regressor | 0.08452272415161133 | 0.021634244918823244 |
| 5 | Bagged decision tree regressor | 0.03954074382781982 | 0.0012121200561523438 |
| 6 | DummyRegressor | 0.003361225128173828 | 0.00020041465759277344 |

* Learning curve comparison
  + Learning curves are plots of the model's performance on the training set and the validation set as a function of varying samples of training dataset.
  + To be specific, learning curves show training & validation scores on the y-axis against varying samples of the training dataset on the x-axis.
  + The training & validation scores could be any evaluation metric like MSE, RMSE, etc. on your training and validation sets.
  + Learning curves can be used to understand the bias and variance errors of a model.
  + Here we are generating learning curves by plotting RMSE on y-axis and Number of training examples on x-axis for all models.
  + For models like KNearestNeighborsRegressor, LinearRegression, and Support Vector Machine regressor; rmse value is decreasing as number of examples are increasing during training.
  + For DecisionTreeRegressor, rmse value drop still the number of training examples were around 825 and it started increasing again from that point still number of training examples reached to around 1100.
  + For Bagged decision tree regressor and DummyRegressor model, we can observe that rmse value was decreasing for some number of training examples, and then it started increasing again.
  + If we observed carefully rmse value of last point learning curves; the LinearRegression and Bagged decision tree regressor have least rmse value respectively.
  + Whereas DummyRegressor has highest rmse value at last point of learning curve.
* rmse comparison with full training data
  + Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.
  + The LinearRegression model is the best performing model among all. Since rmse value is 761.08 and it is less than remaining models.
  + Whereas DummyRegressor and Support Vector Machine regressor models are performed worst considering rmse values.
* Computational time comparison
  + Let’s display two tables for all models ordered from ascending to descending values of their training times and testing times.
  + Training times table:

|  |  |
| --- | --- |
| Model Name | Training times |
| LinearRegression | 0.0028255224227905274 |
| DummyRegressor | 0.003361225128173828 |
| Bagged decision tree regressor | 0.03954074382781982 |
| Support Vector Machine regressor | 0.08452272415161133 |
| KNearestNeighborsRegressor | 0.14352903366088868 |
| DecisionTreeRegressor | 0.24312202930450438 |

* + Testing times table:

|  |  |
| --- | --- |
| Model Name | Testing times |
| DummyRegressor | 0.00020041465759277344 |
| DecisionTreeRegressor | 0.0008022308349609375 |
| LinearRegression | 0.0011689901351928712 |
| Bagged decision tree regressor | 0.0012121200561523438 |
| KNearestNeighborsRegressor | 0.0018187522888183593 |
| Support Vector Machine regressor | 0.021634244918823244 |

* + From training times table, we can say that LinearRegression model takes least time, whereas DecisionTreeRegressor takes most time for training.
  + From testing times table, we can say that DummyRegressor model takes least time, whereas SupportVectorMachineRegressor model takes most time for testing among all given models.