**MINOR PROJECT 2**

**EVEN 2021**

**Project Report (End Evaluation)**

**OLD  CAR PRICE PREDICTION**

DEPARTMENT OF CSE&IT

SUBMITTED BY:         NAME                                                  ENROLLMENT NO

UNDER THE SUPERVISION OF:

SUBMITTED TO:

* Regression Analysis for Old Car Price Prediction
  + 1.1 Introduction
  + 1.2 Tools
  + 1.3 Old car price prediction problem
* 2. Notebook structure
* 3. Methodology
  + 3.1 ELK Stack Analysis
    - Logstash
    - [Elasticsearch](https://github.com/francescopisu/UsedCarPricePrediction#elasticsearch)
    - [Kibana](https://github.com/francescopisu/UsedCarPricePrediction#kibana)
  + 3.2 Regression Analysis
    - Data Analysis
    - 3.2.1 Data preprocessing
      * Removing Outliers
      * Managing categorical attributes
* 4. Comparing regression models
  + 4.1 Parallelizing Hyperparameter Tuning with SparkSklearn
  + 4.2 Linear Regression
  + 4.3 Random Forest Regression
* 5. Experimental Results
* 6. Conclusions

**1.1 Introduction**

This project aims to solve the problem of predicting the price of a used car, using Sklearn's supervised machine learning techniques integrated with Spark-Sklearn library. It is clearly a regression problem and predictions are carried out on dataset of used car sales in the american car market. Several regression tecniques have been studied, including Linear Regression, Decision Trees and Random forests of decision trees. Their performances were compared in order to determine which one works best with out dataset.

**1.2 Tools**

**Software Requirements**

The software requirements in this project include:

1. **Python**

Python is used for creating a backbone structure. Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, it uses whitespace indentation, rather than curly braces or keywords. Python has a large standard library, commonly cited as one of Python's greatest Strengths

**Machine Learning tool - Scikit-learn (Python Package)**

Scikit-learn [9] is an open-source, BSD-licensed, Python library providing simple and efficient tools for data mining and data analysis. Project is built on NumPy, SciPy, and matplotlib. Scikit-learn implements a range of machine learning, preprocessing, cross-validation and visualization algorithms.

Scikit-learn can perform classification (identifying to which category an object belongs to), regression (predicting a continuous-valued attribute associated with an object), clustering (automatic grouping of similar objects into sets.), dimensionality reduction (reducing the number of random variables to consider), Model selection (comparing, validating and choosing parameters and models.) and preprocessing (feature extraction and normalization).

In general, a learning problem considers a set of n samples of data and try to predict properties

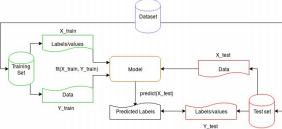
of unknown data. If each sample is more than a single number, and for instance a multidimensional entry (aka multivariate data), is it said to have several attributes, or features. We can separate learning problems in a few large categories:

• **Supervised learning**, in which the data comes with additional attributes that we want to predict. Classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of classification problem would be the digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories.

Regression: if the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.

• **Unsupervised learning**, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

Depending if we perform classification or regression we refer to the classes as labels or values respectively. In classification and regression Scikit-learn follows the schema shown in Figure.



The first step is to split the original dataset into training and test sets, then using the training set we can create a model to predict new values or labels. We predict new values (or labels) using the model previously created and the test data set. Finally, we can compare the predicted labels and the expected labels to measure classifier accuracy.

**Hardware Requirements**

**Linux**: GNOME or KDE desktop GNU C Library (Glibc) 2.15 or later, 2 GB RAM minimum, 4 GB RAM recommended,

**Windows:** Windows 10 2 GB RAM minimum, 4 GB RAM recommended,

**Supportive Systems**

The supported Operating Systems for a client include Windows and Linux. Windows and Linux are two of the operating systems that will support comparative website. Since Linux is an open-source operating system, which we will use in this project is developed on the windows platform but is made compatible with linux too. Anaconda is used as a open-source

The Jupyter Notebook is used as an open-source web application that allows us to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more

Most of the project has been developed using Python as the programming language of choice and the following libraries:

* [Scikit-Learn](https://scikit-learn.org/stable/), regression models and cross validation techniques.
* [Spark-Sklearn](https://github.com/databricks/spark-sklearn), parallelization of the hyperparameter tuning process.
* [Pandas](https://pandas.pydata.org/), data analysis purposes.
* [ELK Stack](https://www.elastic.co/elk-stack), data analysis too.
* [Rfpimp](https://github.com/parrt/random-forest-importances), feature importances in random forests.

**1.3 Used car price prediction problem**

Old car price prediction problem has a certain value because different studies show that the market of used cars is destined to a continuous growth in the short term. In fact, leasing cars is now a common practice through which it is possible to get get hold of a car by paying a fixed sum for an agreed number of months rather than buying it in its entirety. Once leasing period is over, it is possible to buy the car by paying the residual value, i.e. at the **expected** resale price. It is therefore in the interest of vendors to be able to predict this value with a certain degree of accuracy, since if this value is initially underestimated, the installment will be higher for the customer which will most likely opt for another dealership. It is therefore clear that the price prediction of used cars has a high commercial value, especially in developed countries where the economy of *leasing* has a certain volume.  
This problem, however, is not easy to solve as the car's value depends on many factor including year of registration, manufacturer, present\_price, kms\_driven and several other specific informations such as type of fuel type of change (manual, automati), number of doors, number of previous owners, if it was previously owned by a private individual or by a company and the prestige of the manufacturer.  
Unfortunately, only a small part of this information is available and therefore it is very important to relate the results obtained in terms of accuracy to the features available for the analysis. Moreover, not all the previously listed features have the same importance, some are more so than others and therefore is essential to identify the most important ones, on which to perform the analysis.  
Since some attributes of the dataset aren't relevant to our analysis, they have been discarded; so, as mentioned above, this fact must be taken into account when conclusions on the accuracy are drawn.

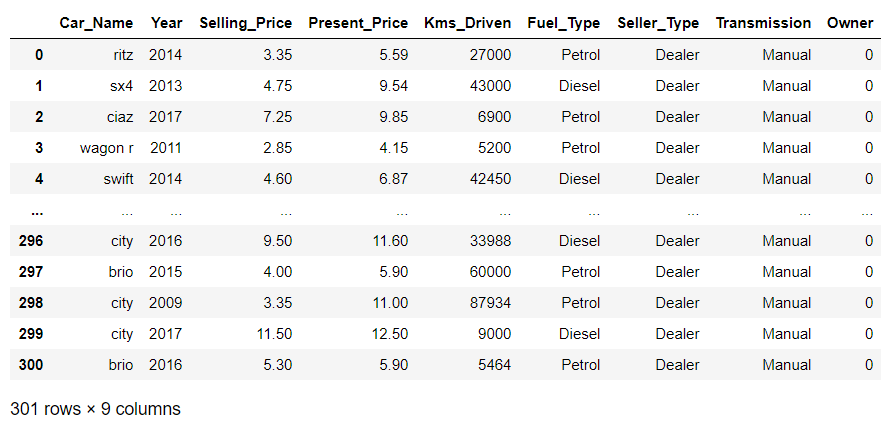
**2 Notebook structure**

The python notebook is structured as follows:

Notebook structure

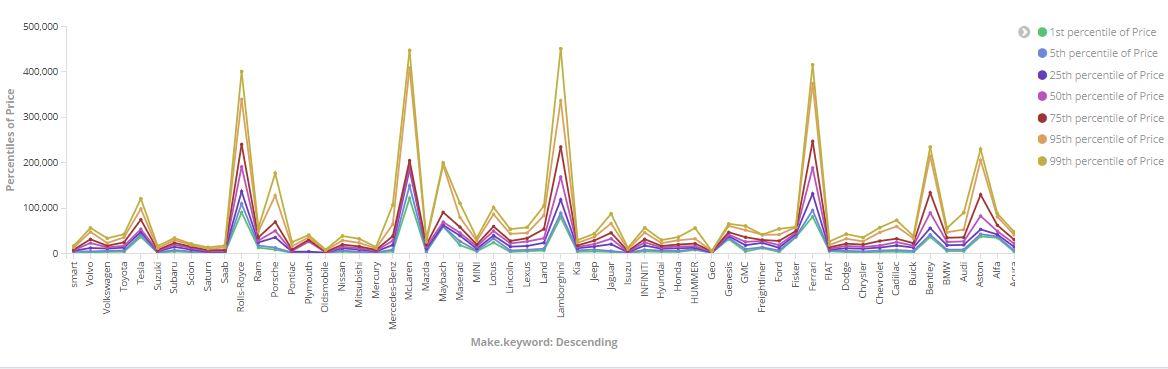
**3 Methodology**

This chapter provides an in-depth description of the followed methodology for solving the problem discussed above, with particular emphasis on the first phase concerning the dataset analysis carried out with ELK stack and the consequent preprocessing of data, followed by the definition, training and evaluation of the chosen regression models, highlighting the importance of integrating these techniques with Spark to parallelize the process of hyperparameter tuning of decision models.  
The [dataset](https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho?select=car+data.csv) on which the regression analysis was performed and Old car sales in the Indian market acquired via scraping on [cardekho.com](https://www.cardekho.com/) car sales portal.  
Each record has the following features: Car\_name,Selling\_ price, Present\_Year, company, Kms\_Driven, Fule\_Type, Transmmiton, Owner\_Type, Seller\_type, fuel\_Type.



**3.1 ELK Stack Analysis**

The **ELK Stack** helps by providing users with a powerful platform that collects and processes data from multiple data sources, stores that data in one centralized data store that can scale as data grows, and that provides a set of tools to **analyze** the data. Of course, the **ELK Stack** is open source. The stack is composed by three tools:**Beats,** **Elasticsearch**, **Logstash** e **Kibana**.

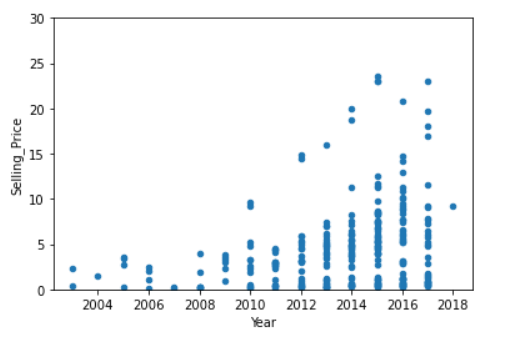


**3.2 Regression Analysis**

Formally, a regression analysis consists of a series of statistical processes aimed at estimating the relationships existing between a set of variables; in particular we try to estimate the relationship between a special variable called dependent (in our case the price) and the remaining independent variables (the other features). This analysis makes it possible to understand how the value of the dependent variable changes as the value of any of the independent variables changes, keeping the others fixed.  
To carry out the prediction, various techniques have been studied including linear regression.

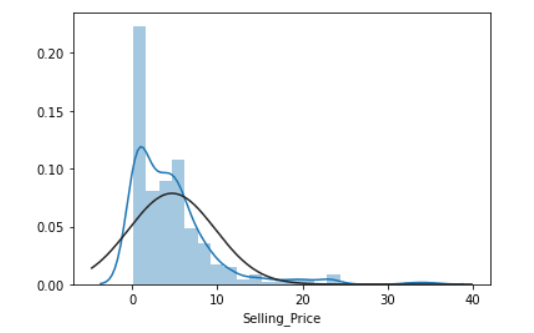
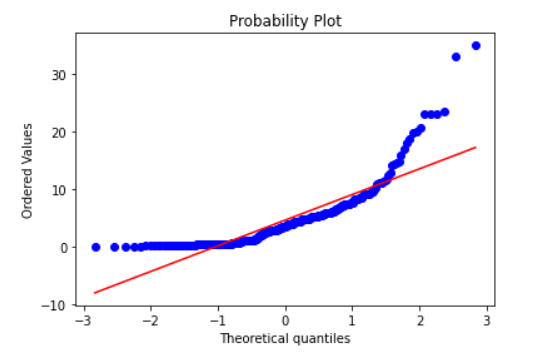
**Data Analysis**

Before preprocessing the data we must take a look to how the dataset shows up. In particular we carry out an analysis on the price attribute: describing it allows us to appreciate some informations such as min and max values and standard deviation. We then proceed to compute skewness and kurtosis of the distribution. Next, we observe its relationship with numerical and categorical features by plotting some graphs.  
Feature importance computation showed us that Year and Selling\_Price are both important for Price attribute and through correlation matrices we can learn a bit more about that.



**3.2.1 Data Preprocessing**

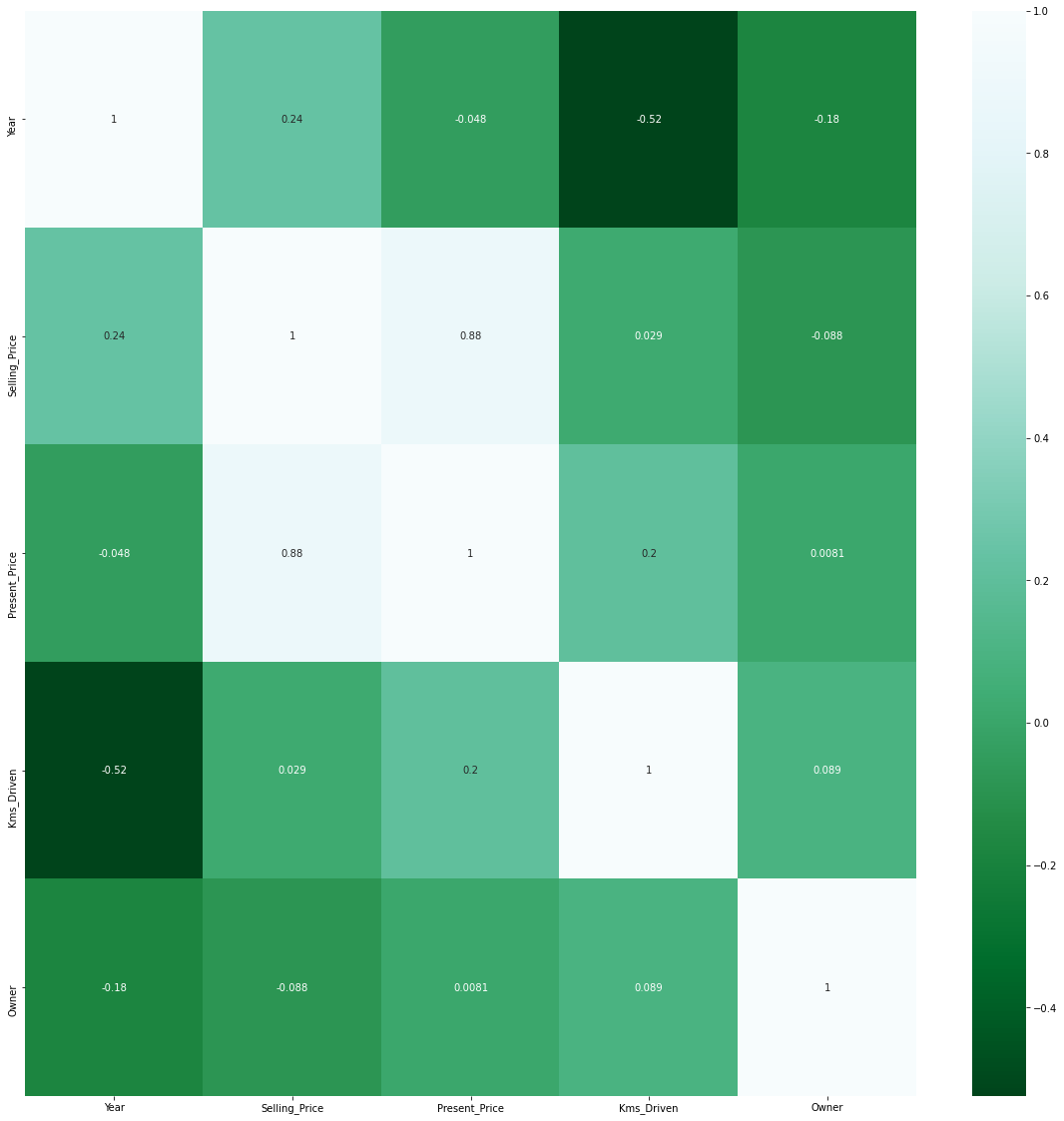
The dataset used to carry out the analysis is one of the best available in terms of cleanliness. Despite this, it was necessary to perform preprocessing in order to minimize the probability of incorrect learning by the models.  
First it was ascertained that none of the attributes of the dataset presented null values; surprisingly, no feature presented null values, so no action was required to do so. Subsequently, the plausibility of the values for each of the numerical attributes (Selling\_Price, Present\_Price, Kms\_Driven) present in the dataset was verified.

**Removing Outliers**

As the Car Manufacturer/Price box plot shows, there is a high presence of outliers in the dataset and the only way to tackle this problem is to apply an outliers removal procedure based on the single car model. To do so, we take only the values between the 20th and 80th percentile of the gaussian distribution; this procedure is then applied to each of 3k models.

**Correlation Matrix:** A Correlation matrix shows the correlation coefficients among independent variables. According to our matrix the following variables shows high correlation with each other:



**Managing categorical attributes**

For managing categorical attributes two different approaches has been taken, depending on the particular regression model.  
For linear regression we had to apply a *One Hot Encoding*  procedure, through which a noolean attribute is added to the dataset for each unique value of the categorical attributes. Clearly, this procedure must be done with caution because the dimensionality of the set ramps up very quickly. In our case the only two categorical attributes were Year and Selling\_Price is a categorical variable with more than 300 values and, as mentioned, OHencoding it will generate a dataset with more or less 300 attributes. That is the technique of choice for linear regressors.  
For decision tree and random forests we simply applied a label encoding procedure, through which an increasing number is associated to each value of a categorical attribute. Label Encoding doesn't fit linear regression well because this type of model will try to find a correlation between those values:

**4 Comparing regression models**

In this section we compare the different regression models used in the analysis.

**4.1 Parallelizing Hyperparameter Tuning with SparkSklearn**

Each of the three prediction models used is characterized by a certain number of parameters (fit\_intercept, normalize and copy\_X for the linear regressor, max\_depth for random forest and decision trees). Based on the value assumed by these parameters, the model may have better / worse performance. The process by which the optimal value is determined for each parameter relative to the training set is called **Hyperparameter Tuning**.  
This is a resource and time-consuming process. The entire project was implemented by exploiting the functionalities offered by the Sci-Kit Learn library (Sklearn) of Python so the parameter tuning process is performed by a function called GridSearchCV: this method performs the fit with every possible combination of parameter values specified in a grid of parameters, executing at the same time the Cross-Validation process that allows to make the most of the available data and reduce the probability of overfitting the model.  
To speed up this process, the tuning phase has been carried out exploiting the *Spark-Sklearn* implementation of GridSearchCV, which uses Apache Spark to parallelize the calculation.

**4.2 Linear Regression**

GridSearchCV for Linear Regression applied to the training set outputs these as the best parameters:

LinearRegression(copy\_X=True, fit\_intercept=False, n\_jobs=1, normalize=True)

We then fit a linear regressor with these parameters. In the table below we can observe several information such as scores obtained on training/test sets, Best Score with CV, R2 score and RMSEs.

**4.3 Random Forest Regression**

This time we can see that the best value for **max\_depth** is around 18 and to be sure we use *rf\_random = RandomizedSearchCV(estimator = rf, param\_distributions = random\_grid,scoring='neg\_mean\_squared\_error', n\_iter = 10, cv = 5, verbose=2, random\_state=42, n\_jobs = 1)* to determine the best value.

**5 Model & Results**

The table below shows experimental results obtained for each model.

| **Regression Model** | **R2 Score** |
| --- | --- |
| Linear | 0.85 |
| Random Forest | 0.91 |

It seems that Linear Regressor achieved the best results since training and test score are very similar to each other and so are RMSEs for training and test. When training's RMSE is much higher than the test one, the model is said to be overfitted. We can see this situation in both decision tree and random forest. Furthermore, linear and random forest regressor's performances are comparable in terms of score, with a small advantage in the best score with CV by Random Forest.  
Declaring linear regressor as the winner would be too hasty from us, also because we must not forget that linear regressor in trained with a dataset composed by more Note that these results have been obtained in a certain cross validation session and each of them will produce slightly different results due to the way the dataset is divided in KFold CV method.

**6 Conclusions**

As we can see in the cross validation scores table, linear regressor and random forest are the ones that perform better, with a training RMSE   
Saying that one model is objectively better than another is difficult, especially in this situation where linear regressor is working on a OHencoded dataset and random forest regressor on a label encoded one. Random forests are almost always preferable to linear regressors because they don't need much preprocessing and sometimes they produce good results even in presence of outliers. In our case the differences in performance between linear regressor and random forest are not enough to justify the exaggeratedly high number of attributes introduces by one hot encoding.  
Furthermore, random forest regressor fits the data in a fraction of the time required by linear regressor as we can see in the data below:

Linear regressor fit:

CPU times: user 31.7 s, sys: 20.2 s, total: 51.9 s

Wall time: 26.3 s

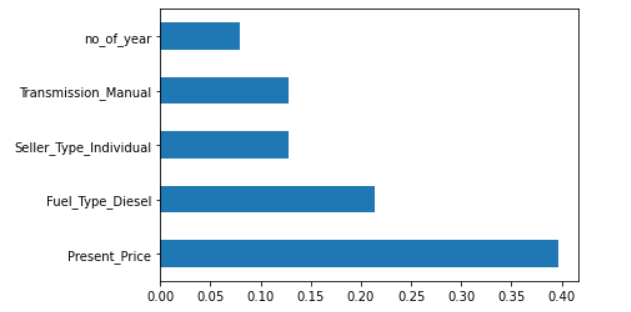
Random forest regressor fit:

CPU times: user 9.31 s, sys: 9.99 ms, total: 9.32 s

Wall time: 9.33 s

Therefore, Random Forest Regressor has been chosen as the final mode. Following the feature importance computed with K=5 KFold Cross Validaion, scores on the hold-out test set and the final RMSE computed on the entire dataset.

**Feature Importance**



**Scores**

| **Regression Model** | **Test Set Score** | **R2 Score** | **RMSE Test** |
| --- | --- | --- | --- |
| Random Forest | 0.948 | 0.911 | 2.00018 |

References:

1. <https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho?select=car+data.csv>
2. N. Monburinon, P. Chertchom, T. Kaewkiriya, S. Rungpheung, S. Buya and P. Boonpou, "Prediction of prices for used car by using regression models," 2018 5th International Conference on Business and Industrial Research (ICBIR) , Bangkok, 2018, pp. 115-119.

3. Listiani M. 2009. Support Vector Regression Analysis for Price Prediction in a Car Leasing Application. Master Thesis. Hamburg University of Technology

4. Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings ofthe 22nd acm sigkdd international conference on knowledge discovery and data mining . ACM, 2016.

5. <https://scikit-learn.org/stable/modules/classes.html> : Scikit-learn: Machine Learning in Python ,Pedregosa et al. , JMLR 12, pp. 2825-2830, 2011