PROJECT SPECIFICATION

Machine Learning Capstone Project

Definition

This project is based on data collected from homes in suburbs of Boston, Massachusetts. A model has been created with training and testing of data set and determined a suitable performance metric for this problem. evaluate the performance and predictive of the model based on this data that is seen as a good fit then it could make predictions about a home –in particular, its monetary value and further more will provide the a model that generalizes for unseen data .This model will work like a brokerage agent of real estate, which can be used the information on the daily basis.

Analysis

The dataset for this project originates from the UCI Machine Learning Repository. The Boston housing data was collected in 1978 and each of the 506 entries represent aggregated data about 14 features for homes from various suburbs in Boston, Massachusetts. For the purposes of this project, the following preprocessing steps have been made to the dataset:

- 16 data points have an 'MEDV' value of 50.0. These data points likely contain **missing or censored values** and have been removed.
- 1 data point has an 'RM' value of 8.78. This data point can be considered an **outlier** and has been removed.
- The features 'RM', 'LSTAT', 'PTRATIO', and 'MEDV' are essential. The remaining **non-relevant features** have been excluded.
- The feature 'MEDV' has been **multiplicatively scaled** to account for 35 years of market inflation.

Methodology

I will develop the tools and techniques necessary for a model to make a prediction. Being able to make accurate evaluations of each model's performance through the use of these tools and techniques helps to greatly reinforce the confidence in your predictions.
Evaluation Matrices:
It is difficult to measure the quality of a given model without quantifying its performance over training and testing. This is typically done using some type of performance metric, whether it is through calculating some type of error, the goodness of fit, or some other useful measurement. For this project,will be calculating the *coefficient of determination*, R2, to quantify the model's performance. The coefficient of determination for a model is a useful statistic in regression analysis, as it often describes how "good" that model is at making predictions.

The values for R2 range from 0 to 1, which captures the percentage of squared correlation between the predicted and actual values of the **target variable**. A model with an R2 of 0 is no better than a model that always predicts the *mean* of the target variable, whereas a model with an R2 of 1 perfectly predicts the target variable. Any value between 0 and 1 indicates what percentage of the target variable, using this model, can be explained by the **features**. *A model can be given a negative R2 as well, which indicates that the model is* **arbitrarily worse** *than one that always predicts the mean of the target variable.*

If we take a close look at the graph with the max depth of 3:

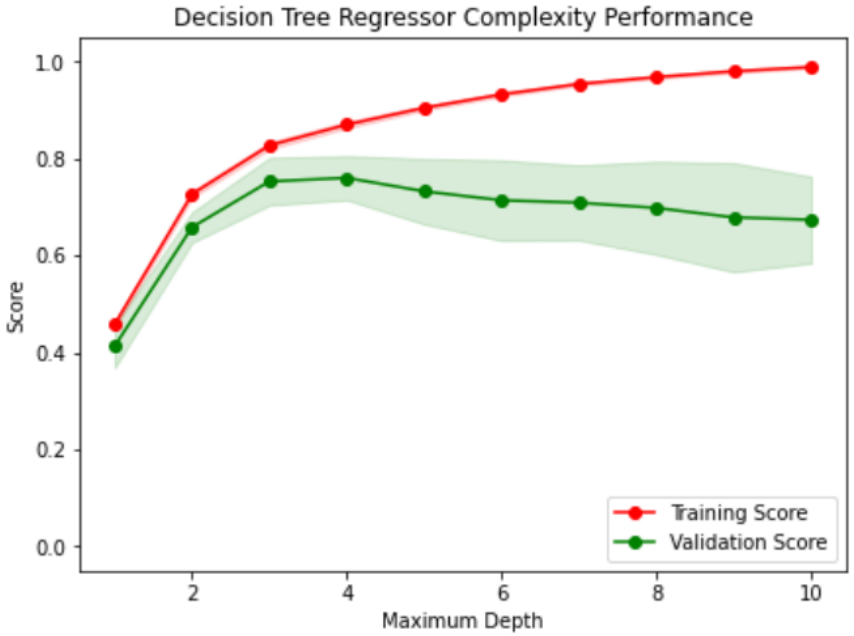
As the number of training points increases, the training score decreases. In contrast, the test score increases.

As both scores (training and testing) tend to converge, from the 300 points threshold, having more training points will not benefit the model.

(Extra question): In general, with more columns for each observation, we'll get more information and the model will be able to learn better from the dataset and therefore, make better predictions.

Four graphs for a decision tree model with different maximum depths. Each graph visualizes the learning curves of the model for both training and testing as the size of the training set is increased. Note that the shaded region of a learning curve denotes the uncertainty of that curve (measured as the standard deviation). The model is scored on both the training and testing sets using R2, the coefficient of determination

A graph for a decision tree model that has been trained and validated on the training data using different maximum depths. The graph produces two complexity curves — one for training and one for validation. Similar to the **learning curves**, the shaded regions of both the complexity curves denote the uncertainty in those curves, and the model is scored on both the training and validation sets using the performance metric function.



Fitting a model has been done ,the final implementation requires that we bring everything together and train a model using the **decision tree algorithm**. To ensure that we are producing an optimized model, we will train the model using the grid search technique to optimize the 'max_depth' parameter for the decision tree. The 'max_depth' parameter can be thought of as how many questions the decision tree algorithm is allowed to ask about the data before making a prediction. Decision trees are part of a class of algorithms called *supervised learning algorithms*.

If we analize how the bias-variance vary with the maximun depth, we can infer that:

- With the maximun depth of one, the graphic shows that the model does not return good score in neither training nor testing data, which is a symptom of underfitting and so, high bias. To improve performance, we should increase model's complexity, in this case increasing the max_depth hyperparameter to get better results.
- With the maximun depth of ten, the graphic shows that the model learn perfectly well from training data (with a score close to one) and also returns poor results on test data, which is an indicator of overfitting, not being able to generalize well on new data. This is a problem of High Variance. To improve performance, we should decrease the model's complexity, in this case decreasing the max_depth hyperparameter to get better results.

Best-Guess Optimal Model

From the complexity curve, we can infer that the best maximum depth for the model is 4, as it is the one that yields the best validation score.

In addition, for more depth although the training score increases, validation score tends to decrease which is a sign of overfitting.

Grid Search

The grid search technique exhaustively generates candidates from a grid of parameter values specified with the param_grid parameter, which is a dictionary with the values of the hyperparameters to evaluate. One eample can be:

param_grid = [{'C': [1, 10, 100, 1000], 'kernel': ['linear']}, {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},]

In this example, two grids should be explored: one with a linear kernel an C values of [1,10,100,1000], and the second one with an RBF kernel, and the cross product of C values ranging in [1, 10, 100, 1000] and gamma values in [0.001, 0.0001].

When fitting it on a dataset all the possible combinations of parameter values are evaluated and the best combination is retained.

Cross-Validation

K-fold cross-validation is a technique used for making sure that our model is well trained, without using the test set. It consist in splitting data into k partitions of equal size. For each partition i, we train the model on the remaining k-1 parameters and evaluate it on partition i. The final score is the average of the K scores obtained.
When evaluating different hyperparameters for estimators, there is still a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can "leak" into the model and evaluation metrics no longer report on generalization performance. To solve this problem, yet another part of the dataset can be held out as a so-called "validation set": training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.
However, by partitioning the available data into three sets (training, validating and testing sets), we drastically reduce the number of samples which can be used for learning the model, and the resulting model may not be sufficiently well trained (underfitting).
By using k-fold validation we make sure that the model uses all the training data available for tunning the model, it can be computationally expensive but allows to train models even if little data is available.
The main purpose of k-fold validation is to get an unbiased estimate of model generalization on new data.

Result

Once a model has been trained on a given set of data, it can now be used to make predictions on new sets of input data. In the case of a *decision tree regressor*, the model has learned *what the best questions to ask about the input data are*, and can respond with a prediction for the **target variable**. We can use these predictions to gain information about data where the value of the target variable is unknown — such as data the model was not trained on.

We are able to predicting the selling price as example in taken

Feature	Client 1	Client 2	Client 3
Total number of rooms in home	5 rooms	4 rooms	8 rooms
Neighborhood poverty level (as %)	17%	32%	3%
Student-teacher ratio of nearby schools	15-to-1	22-to-1	12-to-1

Able to predict selling price for all 3 clients after implementing the prediction

```
Predicted selling price for Client 1's home: $403,025.00
Predicted selling price for Client 2's home: $237,478.72
Predicted selling price for Client 3's home: $931,636.36
```

The predicted selling prices are:

- For Client 1's home: $403,025.00
- For Client 2's home: $237,478.72
- For Client 3's home: $931,636.36

From above, we obtained the following statistics:

- Minimum price: $105000.0
- Maximum price: $1024800.0
- Mean price: $454342.9447852761
- Median price $438900.0
- Standard deviation of prices: $165340.27765266786

Given this values, we can conclude:

- Selling price for client 3 is near the million dollars, which is near the maximum of the dataset. This is a reasonable price because of its features (8 rooms, very low poverty level and low student-teacher ratio), the house may be in a wealthy neighborhood.

- Selling price for client 2 is the lowest of the three and given its features is reasonable as it is near the minimum of the dataset.
- For client 1, we can see that its features are intermediate between the latter 2, and therefore, its price is quite near the mean and median.

An optimal model is not necessarily a robust model. Sometimes, a model is either too complex or too simple to sufficiently generalize to new data. Sometimes, a model could use a learning algorithm that is not appropriate for the structure of the data given. Other times, the data itself could be too noisy or contain too few samples to allow a model to adequately capture the target variable — i.e., the model is underfitted.