Exploratory Data Analysis Assignment Report

Matei Bejan

December 28^{th}, 2019

1 Overview

The purpose of this project is the study, understanding and use of the four main machine learning models presented in the Exploratory Data Analysis course:

1. The Linear Model.
2. Decision Trees.
3. Random Forest.
4. Boosted Trees.

In order to assess the goodness of the models and to further improve them, I will use both metrics and plots presented during the lectures. To this end, I will employ the mean squared and absolute errors, as well as the adjusted R^2 score, alongside residuals plots and quantile-quantile plots.

I have extended the baseline project a little further and have compared two methods of feature selection, the first using the Random Forest model, while the latter employs a Lasso regressor.

The project has been developed in Python using Google Colab. The main libraries employed are numpy, scipy, pandas, sklearn, xgboost and fancyimpute. The project notebook can be accessed here.

2 Data

The data set I have used for this project is the World Development Indicators published by World Bank and forwarded as a .csv file through kaggle. The data set contains 1346 social and economic indicators for 247 countries and geopolitical groups, collected over 55 years, from 1960 to 2015. Thus, it presents a total of 13585 samples.

Please note that the data set's details given above are determined after the preprocessing step, as the data requires some prior restructuring. The initial data will discussed in **4.1. Restructuring the Data**.

3 Goal

My goal for this project is two-fold.

My first goal is to analyze the performance of the models mentioned above by predicting the Gross Domestic Product growth. Thus, all the methods referenced from now on will inherently be regressors.

My second goal is to compare two different feature selection methods: one based on Random Forest and the other, on Lasso regression.

4 Preprocessing

The preprocessing of our data is divided into 6 steps: Restrucuring, Prior filtering, Missing data filtering, Missing data imputation, Feature selection, Outlier Removal and Normalization.

4.1 Restructuring the Data

As specified earlier, the initial dataset is unsuitable for modelling and thus requires some prior restructuring. As it can be seen in figure 1, the initial data presents one sample per feature per country per year. Moreover, there is a noticeable amount of unnecessary data, namely *CountryCode* and *IndicatorCode*.

	CountryName	CountryCode	IndicatorName	IndicatorCode	Year	Value
0	Arab World	ARB	Adolescent fertility rate (births per 1,000 wo...	SP.ADO.TFRT	1960	1.335609e+02
1	Arab World	ARB	Age dependency ratio (% of working-age populat...	SP.POP.DPND	1960	8.779760e+01
2	Arab World	ARB	Age dependency ratio, old (% of working-age po...	SP.POP.DPND.OL	1960	6.634579e+00
3	Arab World	ARB	Age dependency ratio, young (% of working-age ...	SP.POP.DPND.YG	1960	8.102333e+01
4	Arab World	ARB	Arms exports (SIPRI trend indicator values)	MS.MIL.XPRT.KD	1960	3.000000e+06
5	Arab World	ARB	Arms imports (SIPRI trend indicator values)	MS.MIL.MPRT.KD	1960	5.380000e+08
6	Arab World	ARB	Birth rate, crude (per 1,000 people)	SP.DYN.CBRT.IN	1960	4.769789e+01
7	Arab World	ARB	CO2 emissions (kt)	EN.ATM.CO2E.KT	1960	5.956399e+04
8	Arab World	ARB	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	1960	6.439635e-01
9	Arab World	ARB	CO2 emissions from gaseous fuel consumption (%...	EN.ATM.CO2E.GF.ZS	1960	5.041292e+00

Figure 1: Initial structure of the WDI data set.

My first action was to parse the data set and to group all features by year and country. For this I have dropped the *CountryCode* and *IndicatorCode* columns and added a column for every feature. Next, I have discarded the *Value* column and placed the sample in its respective column. This way each sample will contain all indicators observed in a certain year, as in figure 2.

	CountryName	Adolescent fertility rate (births per 1,000 women ages 15-19)	Age dependency ratio (% of working-age population)	Age dependency ratio, old (% of working-age population)	Age dependency ratio, young (% of working-age population)	Birth rate, crude (per 1,000 people)	CO2 emissions (kt)	CO2 emissions (metric tons per capita)	CO2 emissions from liquid fuel consumption (% of total)	CO2 emissions from liquid fuel consumption (kt)	Death rate, crude (per 1,000 people)
0	Afghanistan	162.7380	101.094930	4.438311	96.656619	49.029	2676.910	0.221827	71.506849	1914.174	15.555
1	Afghanistan	163.3270	101.007981	4.464953	96.543028	48.896	2493.560	0.194971	71.176471	1774.828	15.008
2	Afghanistan	163.9160	100.880590	4.496947	96.383643	48.834	1426.463	0.103776	67.352185	960.754	14.524
3	Afghanistan	164.1812	100.881545	4.537744	96.343801	48.839	1375.125	0.092761	67.200000	924.084	14.101
4	Afghanistan	164.4464	100.847802	4.589312	96.258490	48.898	1320.120	0.083184	67.222222	887.414	13.736
5	Afghanistan	164.7116	100.476266	4.654247	95.822019	48.978	1268.782	0.075646	67.052023	850.744	13.421
6	Afghanistan	164.9768	101.231924	4.662978	96.568946	49.039	1199.109	0.068592	66.972477	803.073	13.146
7	Afghanistan	165.2420	101.976746	4.651200	97.325547	49.036	1114.768	0.061814	66.776316	744.401	12.894
8	Afghanistan	161.4432	102.539894	4.624247	97.915647	48.930	1056.096	0.057051	67.013889	707.731	12.651
9	Afghanistan	157.6444	102.925568	4.588172	98.337396	48.699	832.409	0.043723	60.352423	502.379	12.406

Figure 2: Restructured WDI data in a modelling-compatible format.

The script that handles the restructuring was written separately and can be accessed here. The reason for which I took the decision of separating the two concerns was that, although simple and pretty straight-forward, the restructuring process is time-consuming, as it requires parsing over 5.5 million samples from the initial WDI dataset.

4.2 Filtering irrelevant years

Our data set contains sample that have been observed as late as the 1960s. Even though some features are considered generally valuable for predicting GDP growth, historical data from the 60s and up to the 80s is outdated in respect to the current economical landscape. For this reason I have chosen to filter out all the data from before 1990, year in which a political and economic power vacuum has been created with the fall of the Eastern block.

We have thus shrinked down our current sample count to a bit under half of the initial one: from 13830 samples we only kept 6420.

4.3 Filtering features with high NaN count

Given the nature of the data and the time the observations were made, it is not surprising that, possibly for political or technical reasons, plenty of our 13000 samples have up to 90% missing values. Thus, our WDI data is highly-sparse.

Many of those with over 99% missing data regard children employment and working hours, as well as external debt indicators. Those with a proportion over the 9^{th} decile present economical indicators, such as bonds, incomes and taxes, heath indicators, such as mortality rates for infants and females, education expenditures and logistic indicators regarding electricity and freshwater.

Taking into consideration the high percentage of NaNs, it is natural to believe that many of those indicators bear little usable information. This being the case,

I have decided to set a 40% threshold on the amount of missing values a feature can hold in order to be considered in our further modelling.

The 40% threshold filtering has provided a number of 337 features our of the initial 1346 features.

4.4 Missing data imputation

I have further dealt with the missing data by imputing it using the *kNN* procedure provided by the fancyimpute library.

This approach is based on Beretta and Santaniello's insight on the matter of nearest neighbour imputation. In their paper, the authors suggest that kNN gives an adequate trade-off between the precision of imputation and the ability to preserve the nature of the data when the maximum percentage of missing values lies between 15% and 30%. Furthermore, they note that a value of $k = 3$ has given the best results in respect to the precision-preservation trade-off mentioned above.

After playing around with the k number of neighbors and the missing data percentage, I have observed that the imputation method does not give sub-optimal results when the number of missing values is above 30%. I have therefore applied it directly on our previously-filtered data.

Remark 1: I have opted for using kNN to infer missing values in the detriment of more conservative methods, such as replacing with the mean or the mode, in order to test it and to see how well it performs on highly-sparse data.

Remark 2: Another method which I've taken into consideration for the NaN imputation step was the SoftImpute procedure offered by fancyimpute. However, for this method to converge, a preliminary normalization of the data via fancyimpute's BiScaler procedure. If we applied bi-scaling to our data, there would be no way of re-scaling our data back to its original space.

Remark 3: While playing with these imputation methods, I have combined kNN with SoftImpute. I applied the first on those features with at most 30% missing values and the latter on the rest. After the double-step imputation I have observed high correlation between all features, suggesting that their combination induces a high grade of multiple correlation among our predictors.

4.5 Feature de-correlation

Besides the **annual GDP growth**, our dataset contains another measure of GDP and that is the **annual GDP growth per capita**. It is obvious the two are directly correlated and that it makes little sense to keep both features for the same year, since one can be modeled by using the other.

My approach here, instead of removing the GDP growth per capita feature, was for every sample to use the GDP growth per capita from the previous year. Thus, the 2015 sample with have the 2014 growth per capita and so on.

4.6 Feature selection

The feature selection process is straight-forward and the same for both the Random Forest and the Lasso Regression and it boils down to the following:

1. Split data into predictors and the value to predict. Note that we do not use a train-test split - we consider all samples for training as our goal isn't to use this model to make predictions.

2. Train the model on the data and get the feature scores.

3. Select a number of features with respect to a certain threshold. Note that those are different for our methods.

4. Form a list from the top features selected at the previous step and filter the data using it.

4.6.1 Feature selection via Random Forest

I've modeled a Random Forest model using RandomisedSearchCV, with 5 iterations, and KFold to search over a parameter grid. The search yielded a possible best model which I've fitted on the data and that has scored the top features as it can be seen in figure 3.

Remark: I will come back later to the subject of grid search and k-fold validation, in the actual modelling section.

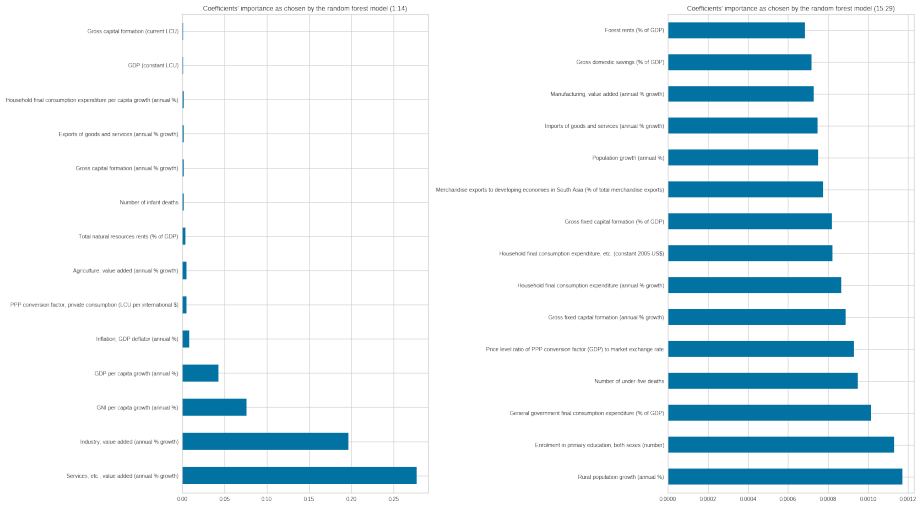

Figure 3: Top 30 indicators chosen by the Random Forest. Note that the right-hand side scale is much smaller than the left-hand side.

As for the number of features to be selected, I've added up the scores until a comfortable *features-to-score* ratio was achieved. This way I decided to use

first 37 features, which yielded a summed score of 0.9665844774408208. I've done some test and it seems that, empirically, this ratio offers the best scores. Adding more features doesn't actually help the models learn much better, while eliminating features worsens the models.

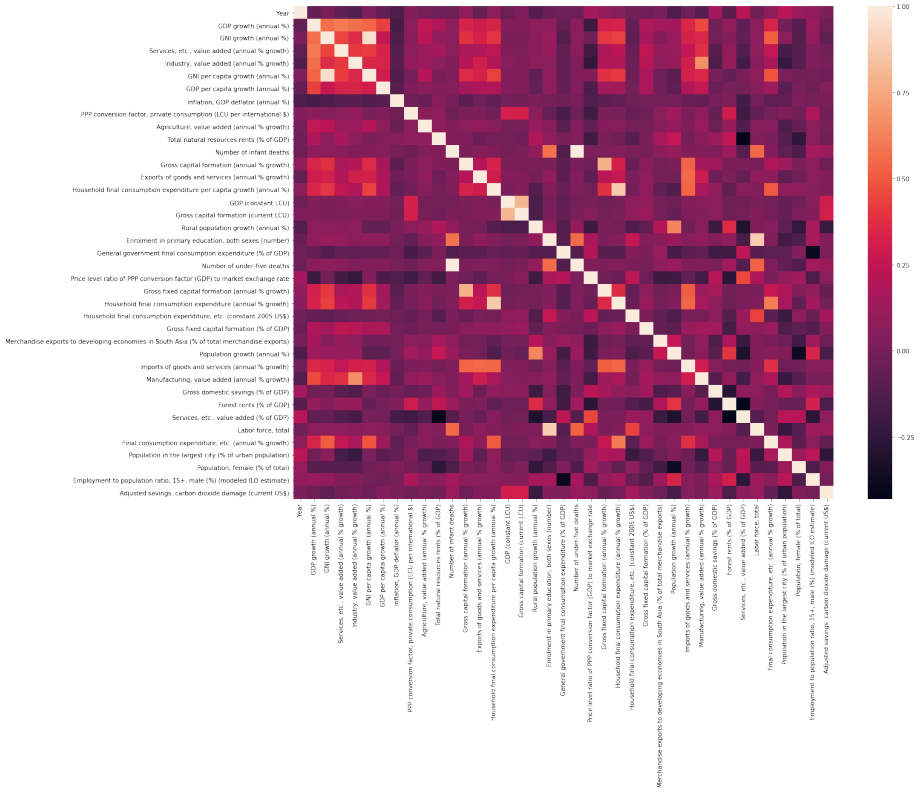

Figure 4: RF-selected features correlation matrix.

4.6.2 Feature selection via Lasso Regression

I have run a cross-validated Lasso Regression on a 70-30 train-test split of the processed WDI data, with 5 folds. For this end I have employed sk-learn's LassoCV, which combines the training and cross validation steps, searching for the optimal penalization cost among 1, 0.1, 0.001, 0.005, 0.0005, with a tolerance of 0.001.

As I mentioned before, the ways of determining the number of features to selects are different for the two models. For Lasso, I used the same amount of the features with the highest scores as with Random Forest.

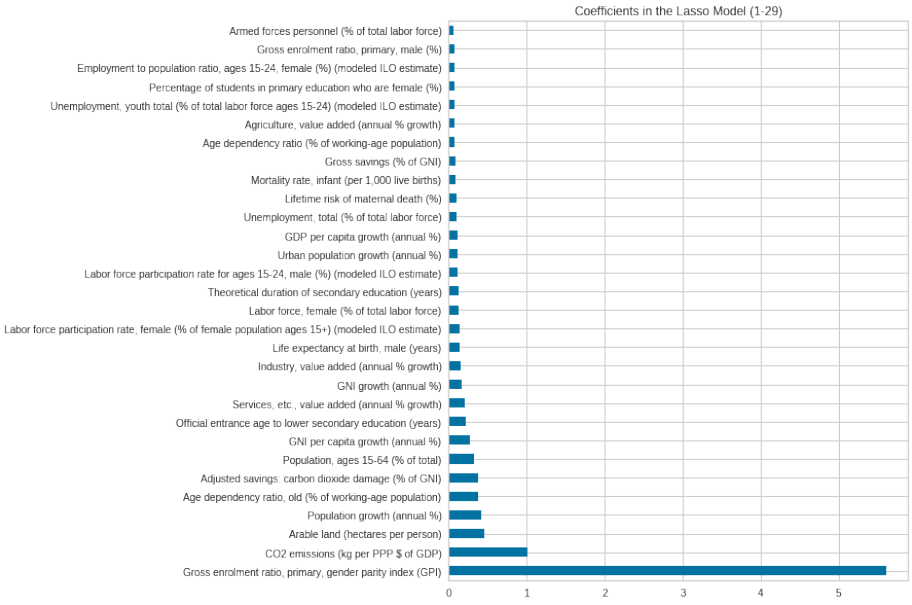

Figure 5: Top 30 indicators chosen by the Lasso Regression.

4.7 Outlier Removal

For outlier removal I used the Z-Score. According to Wikipedia, the Z-score - also called standard score - it's the measure of how many standard deviations below or above the population mean a raw score is. The formula for the Z-Score is:

$$\frac{X - \mu}{\sqrt{\sigma^2/n}},$$

where μ and σ denote the population mean and standard deviation, respectively, X is the sample and n the number of samples.

In a nutshell, the Z-Score gives us an idea of how far from the mean a data point is.

The procedure I used can be summed up in two steps:

1. For each feature, compute the Z-Score of each sample, relative to the mean and standard deviation.

2. Then filter out the sample only is the absolute value of its Z-Score is below a certain the threshold. The threshold I've chosen is 3, as in most of the cases a value of 3 or -3 is used.

This can be done pretty easily in Python using pandas and scipy, as it can be seen in the Colab notebook.

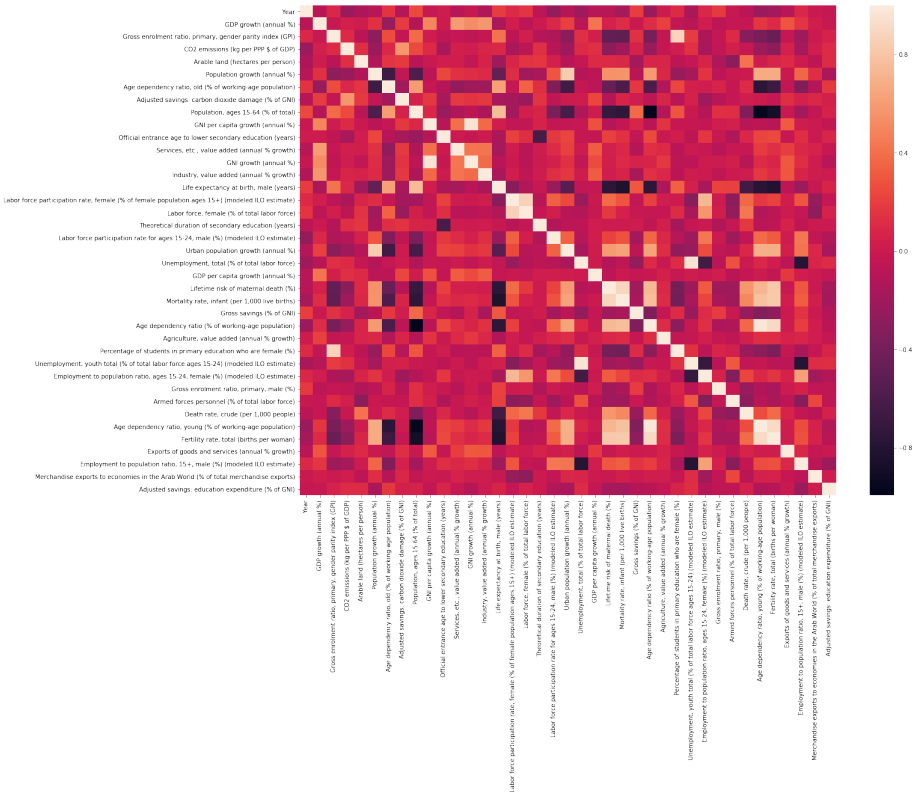

Figure 6: Lasso-selected features correlation matrix. When compared to it's RF-counterpart, one can clearly see a higher degree of correlation among features.

Note: I've also tried to remove the 99^{th} quantile from the data, but got worse results.

4.8 Normalization

Normalization is a common requirement for many machine learning models: they might behave badly if the individual features do not more or less look like standard normally distributed data. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. Not every dataset requires normalization, it is useful only when features have different ranges.

Normalization can be done in multiple ways, either through min-max scaling, standard scaling, unit length scaling etc.

I have chosen to use the standard scaling method via sklearn's StandardScaler. This type of scaling standardises features by removing the mean and scaling to unit variance, such that the data has a mean of 0 and a standard devi-

ation of 1. The reason for choosing this technique over others is it's widespread usage.

5 Modelling with the RF-selected features

The first step in the prediction process is to divide the dataset into train and test data. This has been done by applying a 70-30 train-test split. Please note that the split of data is not automated and is not randomized. I have split the data manually so both the train and test sets are time-wise contiguous. This way, when we run our models, we will be predicting the full GDP growth for each country, from the first recorded year to the last.

I have left all the metric calculation and the plotting for the **Model Comparison** section.

5.1 Linear Model

Even though it is the simplest model out of the four which are included in this report, the linear model yielded better results than expected. This being said, it did not deliver as good as the Random Forest or the Boosted Trees models.

The modelling in this case has been straight-forward, consisting in instantiating the model and fitting it.

5.2 Decision Tree

The second is the Decision Tree. Even though versatile and simple, they are prone to overfitting, usually due to the model's complexity and the Greedy strategy used for splitting does not converge towards a global optimum. Moreover, trees are prone to bias, which is caused either by a dominant feature in the tree or by the way the splits are made. These tendencies will become obvious in the **Model Comparison** section.

I've modeled the Decision Tree by using the RandomisedSearchCV and KFold methods from sklearn, aforementioned in the **Feature selection via Random Forest** subsection.

I started modelling be creating a parameter grid - a dictionary of lists - which the best parameters will be search on. I have then fed this list to the RandomisedSearchCV procedure, with 10 iterations and *negative mean squared error* as scoring metric. Other parameters that I have chosen are the number of concurrent workers, 100, so the search completes relatively fast, and the level of verbosity, 10, in order to trace back any error that might come up.

Another important parameter for the RandomisedSearchCV is the cross-validation option, for which I have employed the K-Fold algorithm with 10 folds. This algorithm consist in splitting the training data into K subsets, train the model on K-1 of them and then test it on the last subset, until all subsets have been used for testing.

Using K-Fold helps reduce both underfitting, since the whole initial training set is used for training, and overfitting, since the whole training set is used for testing.

My choice of K is based on James' and Witten's proposal that the value of K should be either 5 or 10, since these values lead to a healthy bias-variance trade-off.

Remark: I chose the randomized grid search over its exhaustive counterpart because, even though the latter yearns the best set of parameters, the search space scales exponentially with more parameters or more choices for parameters adding to the grid.

Note: I applied the same procedures when modelling the Random Forrest and the XGB models.

The parameters on which I've run the cross-validation are:

1. **max_depth**: Maximum depth of the tree.
2. **min_samples_per_leaf**: The minimum number of samples a leaf node can have.
3. **min_samples_split**: The minimum amount of samples considered for a split.
4. **max_features**: The number of features to consider when looking for the best split.

The following figure contains best set of hyperparameters on which I have trained the Decision Tree model:

```
{'min_samples_split': 152, 'min_samples_leaf': 82, 'max_features': 32, 'max_depth': 185}
```

5.3 Random Forest

The Random Forest regressor has been modeled on a more broad parameter grid. The Random Forest has scored better results than the rpevious models.

1. **bootstrap**: A boolean telling the random forest whether to use bootstrap in order to generate sub-samples on which the trees will be built. If *false*, the whole data set is used to build each tree.
2. **n_estimators**: The number of trees in the forest.

The following figure contains best set of hyperparameters on which I have trained the Random Forest model:

```
{'n_estimators': 1450, 'min_samples_split': 52, 'min_samples_leaf': 7, 'max_features': 32, 'max_depth': 460, 'bootstrap': True}
```

5.4 XGBoost

In this report the Boosted Trees have been modeled via the XGBoost library. Even when compared to the Random Forest, XGBoost has many parameters, which can be grouped as follows:

1. **General parameters** relate to which booster we are using to do boosting, commonly tree or linear model
2. **Booster parameters** depend on which booster you have chosen
3. **Learning task** parameters decide on the learning scenario. For example, regression tasks may use different parameters with ranking tasks.

The real challenge when modelling the Boosted Trees regressor was the research and the understanding of the XGBoost library, one of, if not the most complex gradient boosted trees libraries. Its complexity level is given by the sheer number of hyperparameter it has. I've decided to not include their meaning in this report as the documentation is explicit enough when it comes to their description.

The following figure contains best set of hyperparameters on which I have trained the Random Forest model:

```
{'tree_method': 'exact', 'subsample': 0.6, 'objective': 'reg:squarederror', 'max_leaves': 4, 'max_depth': 820, 'max_delta_step': 8,
'gamma': 6.0, 'eval_metric': 'rmse', 'eta': 0.0005, 'colsample_bytree': 0.64, 'colsample_bynode': 0.87, 'colsample_bylevel': 0.63,'booster': 'dart'}
```

5.5 Model Comparison

Figure 7 encapsulates all metrics calculated for the four models.

	Maximum Residual Error	Mean Squared Error	Mean Absolute Error	Adjusted R^2	R^2	Explained Variance
Linear Model	4.629410	0.256380	0.317761	0.754833	0.757224	0.757224
Decision Tree	4.535511	0.429461	0.445253	0.589323	0.593327	0.593348
Random Forest	4.518405	0.262827	0.308816	0.748669	0.751120	0.751465
XGBoost	4.703513	0.260729	0.324029	0.750675	0.753106	0.753280

Figure 7: Metrics comparison for our four models.

Alongside the metrics presented during the Exploratory Data Analysis lectures, I have included two additional metrics:

1. **Maximum Residual Error** is the largest prediction error in our data.

2. **Explained Variance** measures the proportion to which the model accounts for the variance of a given data set. A fraction of explained variance equals the squared correlation coefficient.

The Linear Model has obtained slightly better scores than the Random Forest or the XGB models. Although it is a bit unexpected, the margin between the scores is a small one. These results are however counter-intuitive as in practice, as we will see in the **Real vs Predicted** plot later, the last two models are more accurate in their predictions. Note that the Linear Model also has the second highest maximum residual error.

As it was to be expected, the Decision Tree model has scored the worse metrics overall. As we will see in our further analysis, this is due to underfitting.

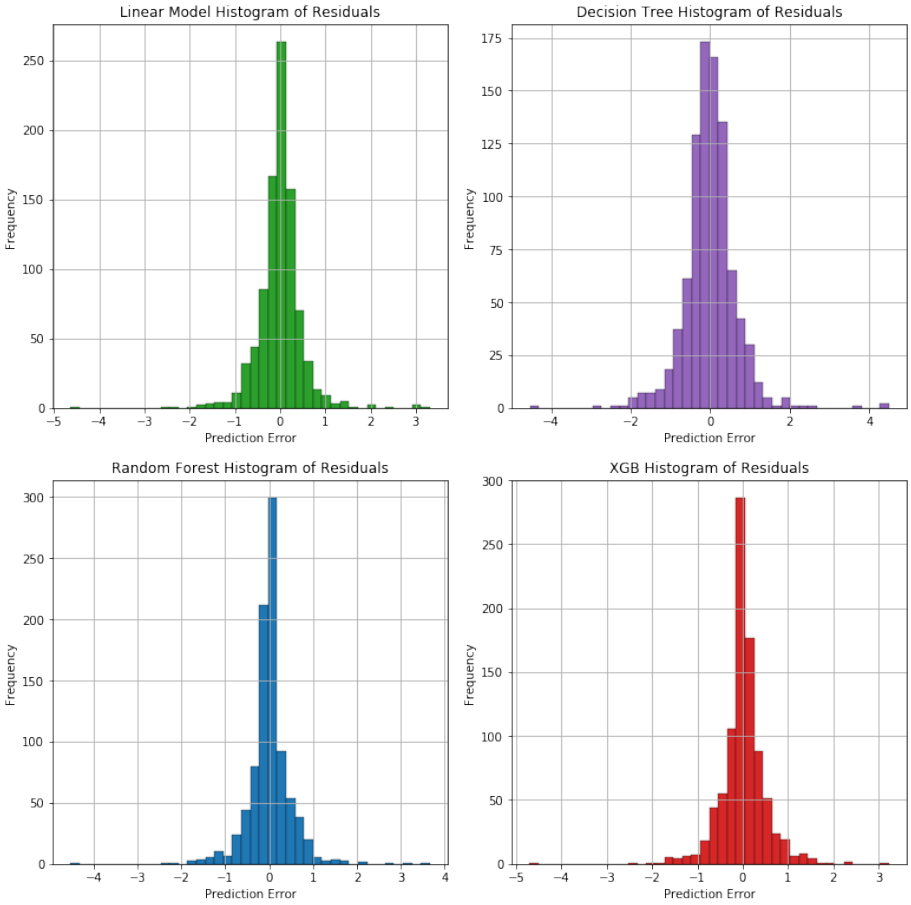

Figure 8: Residuals histograms comparison. From left to right: Linear Model, Decision Tree, Random Forest, XGB.

In figure 8 we can see a high count of residuals close to zero for both the

Random Forest and XGB, as well as a small outlier count. The other two models' residuals follow the normal distribution a bit too closely: they have relatively high prediction errors. This is the case especially for teh Decision Trees, which have a low 0-residuals count and a high proportions of error - almost the same of both positive and negative sides.

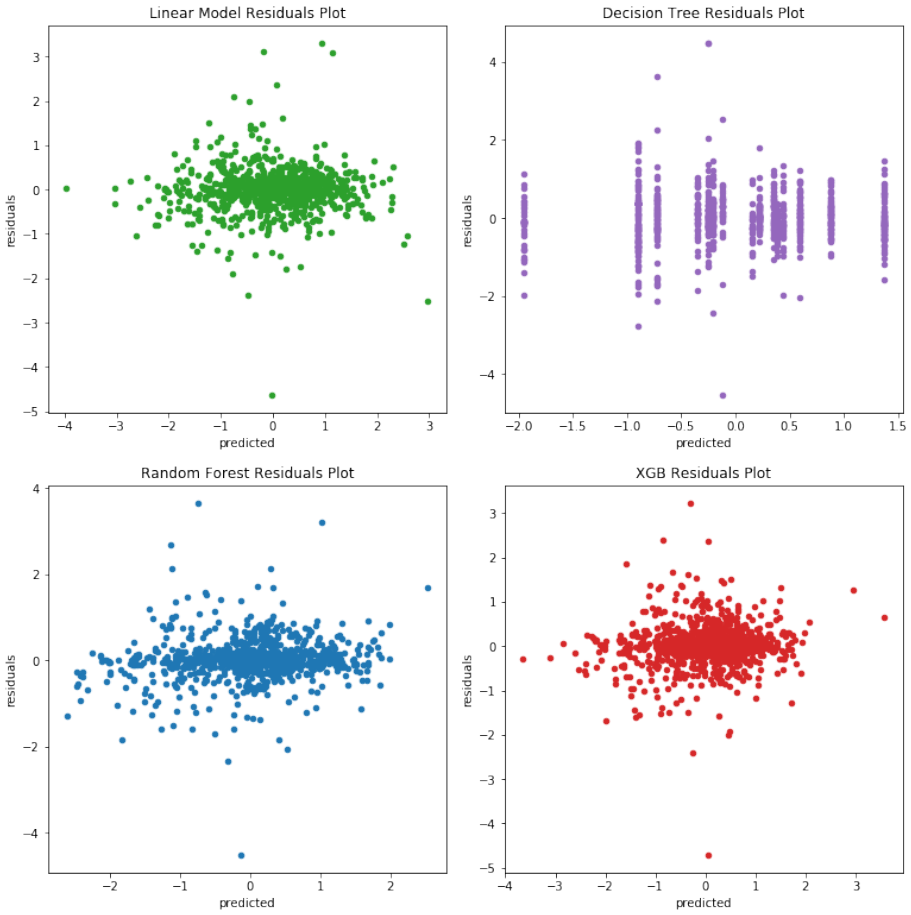

Figure 9: Residuals scatter plots comparison. From left to right: Linear Model, Decision Tree, Random Forest, XGB.

In figure 9 we take a closer look at the residuals through their scatter plots. Once again, the Random Forest and the XGB present a more compact cluster around (0, 0), as well as less residuals. Note that the Random Forest plot is slightly dispersed along the $y = 0$ axis.

The Decicion Trees however, present vertical residuals centered around points that lie on the $y = 0$ axis. This type of behavior is common with binary variables. Moreover, we can clearly see a higher count of outliers in the Decision

Tree's scatter plot, which suggest either an underfit or an overfit.

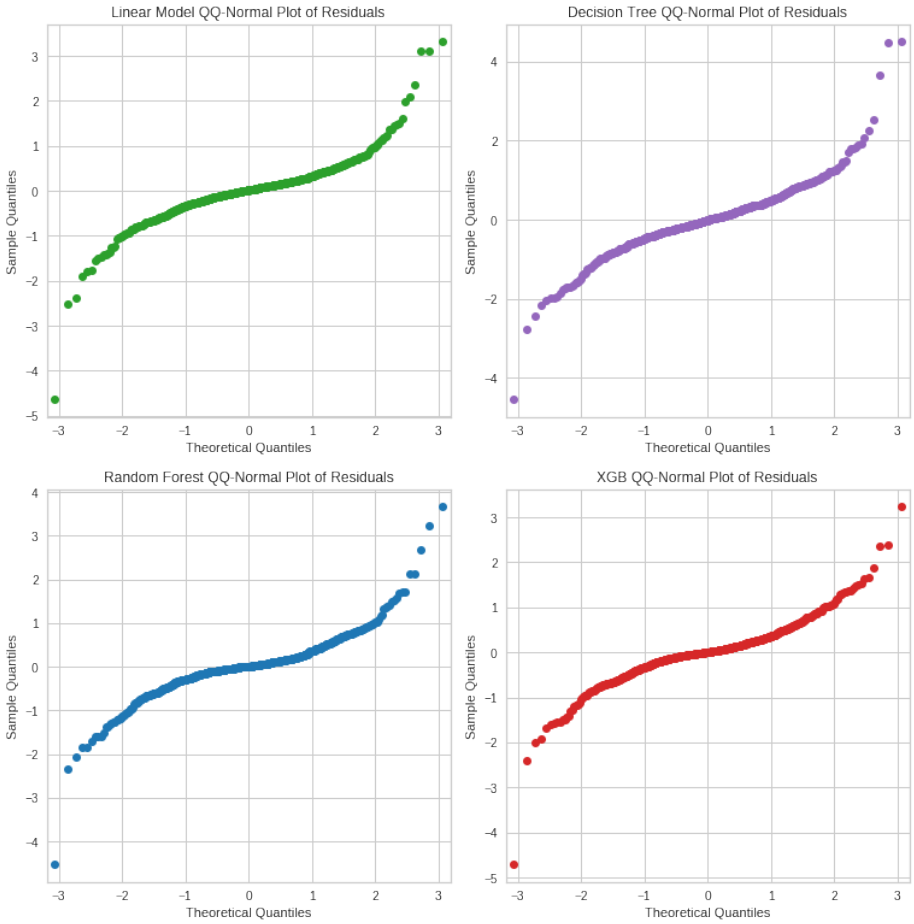

Figure 10: QQ-Normal plots comparison. From left to right: Linear Model, Decision Tree, Random Forest, XGB.

The last residuals comparison plots regard the QQ-Normal plots. The QQ-Normal plot - or the quantile-quantile plot - is determined by plotting the distribution of our residuals against a normal distribution. Its purpose is to help describe our results by comparing it with a normal distribution.

As a reference, $X \sim N(0, 1)$ has a diagonal QQ plot. The QQ, plot, however, can take many shapes, the most frequent being:

1. Right-skewed and Left-skewed plots, in which the majority of the data is located to the right or to the left of the set, respectively.

2. Light-tailed, which suggest that a high portion of our data is located in the extremities.

3. Heavy-tailed, which denotes the preponderance of extreme values.

The Random Forest and XGB models present heavy tailed plots. The "tail" term denotes the tails of the residuals histograms which we've seen in figure 7. The Decision Tree has a light tailed plot, while the Linear Model lies somewhere in between those two types. A characteristic all four plots denote, however, is the presence of extreme outliers, which are found over the ± 3 threshold.

In figure 15 (page 22) we see how our four regressors have performed in respect to the real values.

Even though the Random Forest and the XGB yield relatively good results, there are some interesting cases of underfitting, such as *St. Martin*, with its spikes and relative stagnation in between 2002 and 2010, and the *Virgin Islands*, with their complete stagnation in 1995-2003 and 2007-2010. However, the XGB - and the Random Forest to some extent - capture the 2012 spike trend in the Virgin Islands' case. Meanwhile, there is little to no cohesion between our regressors and the real values in case of St. Mastin.

The Linear Model performs a little more poorly than its ensemble counterparts when it comes to spike estimates, but otherwise its performance its close to the Random Forest and the XGB, as the metrics have suggested.

On the other hand, the Decision Tree rarely gets an accurate trend in our data.

6 Modelling with the Lasso-selected features

All four regressors have been modeled the same way as in the last secion, the only difference being the features, which now have been chosen via Lasso regression.

6.1 Model Comparison

	Maximum Residual Error	Mean Squared Error	Mean Absolute Error	Adjusted R^2	R^2 Score	Explained Variance
Linear Model	4.974216	0.401038	0.379904	0.596532	0.601615	0.601615
Decision Tree	5.013004	0.506221	0.440962	0.490711	0.497128	0.497784
Random Forest	4.966867	0.455465	0.372034	0.541775	0.547548	0.547968
XGBoost	4.709812	0.402479	0.366760	0.595081	0.600183	0.600221

Figure 11: Metrics comparison for our four models.

With a single glance at figure 11 we can see that all scores have worsened. The MSE for the Random Forest is, however, slightly better than the Linear

Model or the XGB. On the other hand, its adjusted R^2 is slightly worse than the other two models.

The Decision Tree is yet again is again the last of the estimators in the order of its estimation accuracy.

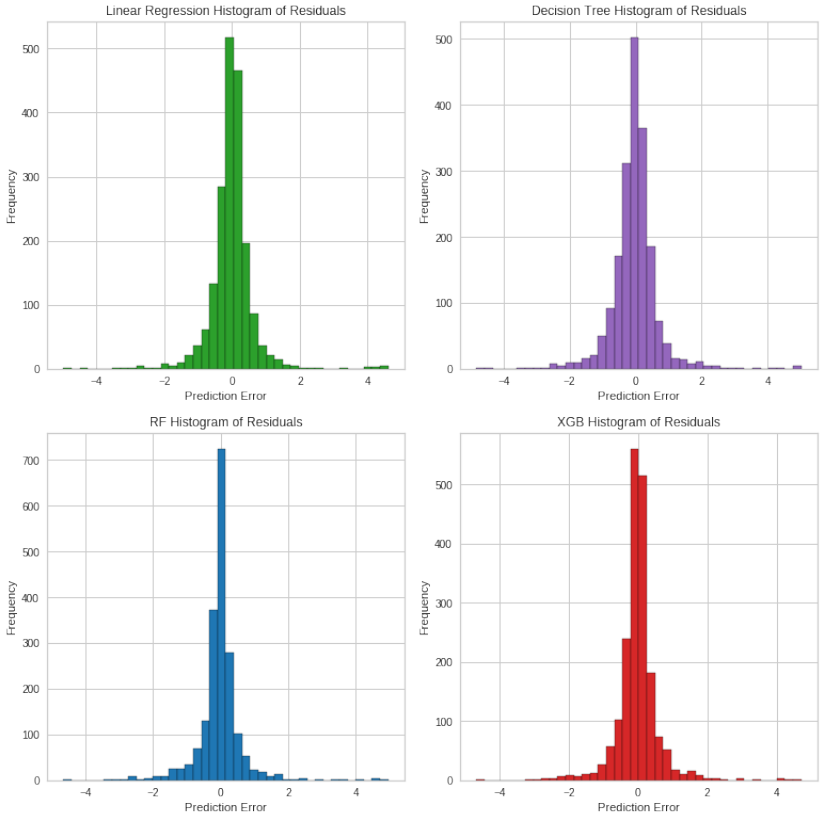

Figure 12: Residuals histograms comparison. From left to right: Linear Model, Decision Tree, Random Forest, XGB.

All models present a higher concentration of 0-value residuals, with the Random Forest topping them all with over 700 such values. However, all distribution have gained a heavier tails, suggesting a higher count of prediction errors. Another new feature in our plots is the prevalence of outliers.

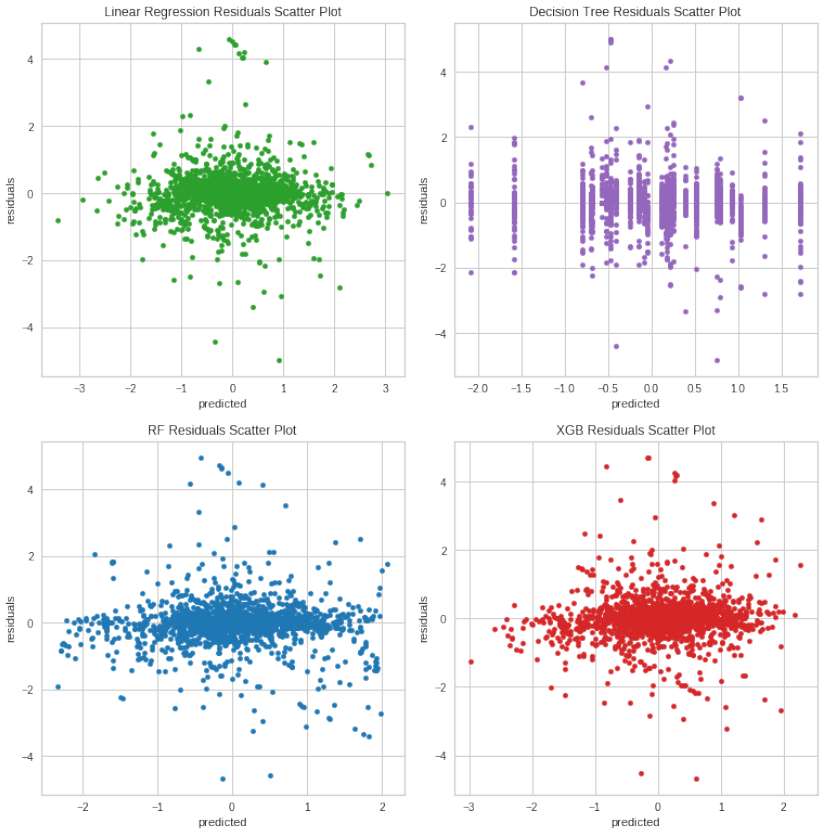

Figure 13: Residuals scatter plots comparison. From left to right: Linear Model, Decision Tree, Random Forest, XGB.

The scatter plots back up the histograms presented earlier by both higher degree of dispersion of points and higher outlier count. Apart from the Decision Tree, all models' residuals now present an elongated shape along the $y = 0$ axis. The former, however tends to keep its shape, with only a higher count of outliers and a stronger tendency towards (0, 0).

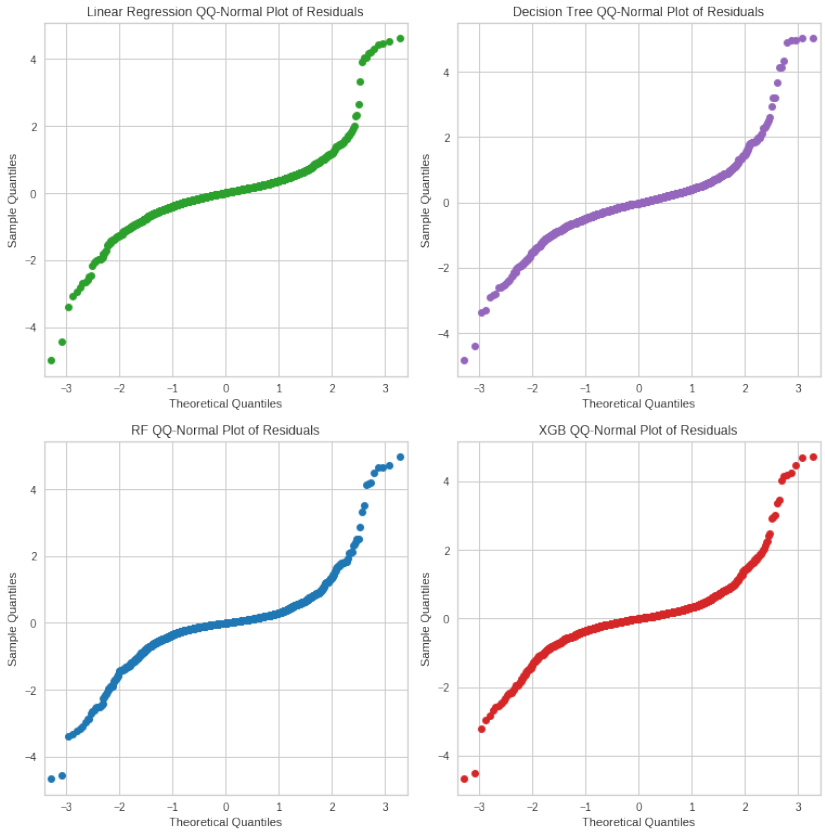

Figure 14: Residuals QQ-Normal plots comparison. From left to right: Linear Model, Decision Tree, Random Forest, XGB.

Last but not least, all QQ-Normal plots have shifted towards a very heavy-tailed form, which is not surprising, given the distribution histograms in figure 11. The S-curbs in the extremities of the plot suggest an increase in the non-linearity of the relationships between our new features. Out all four, the Decision Trees tend to be the ones less affected by the new features. There is also a higher preponderance of outliers, especially on the right-side of the plots, where these points are packed tightly.

As per figure 16 (page 23), we can see first see a tendency towards underfitting when it comes to predicting the growth in between 1990 and 1995, namely the case of Montenegro or Moldova. Furthermore, there seems to be an increase in performance when it comes to countries with relative similar geo-political background, such as Netherlands and Romania. Meanwhile, the estimations for less politically prominent countries, such as St. Martin or the Virgin Islands, become worse.

7 Conclusions

Modelling on the World Development Indicators dataset is hard task, mainly because of the sheer amount of indicators it has. Having over 1300 indicators, a statistical, manual analysis of the possible features was rendered impossible, leaving only the automated feature selection methods.

Surprisingly enough, and a bit counter-intuitive given the non-linearity suggested by the QQ-plots, the Linear Model as been almost as proficient at predicting the GDP growth as the Random Forest or the XGB. However, as it was expected, the Decision Tree has yielded suboptimal results, given the complexity of the task, the number of features and the large number of samples.

Another surprise was the trade-off that Lasso has made in comparison to Random Forest when it comes to feature selection: the results were more accurate in the case of politically and economically prominent countries, but less accurate when it came to lesser countries, economically speaking.

Moreover, the models tend to catch the 2000 and 2010 economical downspikes, with more or less accuracy. This is interesting, since this suggest there is enough information within these indicators and thus there wouldn't a need to employ extra indicators, such as the Baltic Dry Index or the Korean Index.

While doing this project I have toyed with the GDP predictor, transforming its relative values into absolute ones by considering 1990 to be "year zero". This approach did not give better results overall, however. Another test which I've made was to eliminate the *dot com bubble* years, namely 2000 to 2002. Once again, this has given better results than my first trials, when I trained the Random Forest regressor on unstandardized and unfiltered data. These results were nonetheless underperformant when compared to the current approach.

8 Further work

An interesting study would be running a more exhaustive search on the number of features that offer the best trade-off when it comes to model complexity versus its predictive power. Such a search, however, is time consuming if done with tools such as Jupyter notebooks, since they cannot be run as scripts and given the dataset's dimension. My guess would be that, in the case would be that a better number of feature might be between 37 and 45, which would raise the summed importance score from around 0.96 to around 0.97. From my test, even if the summed score passes 0.96 mark, the performance of the regressors decreases quite drastically.

It would also be interesting to use either the Baltic Dry Index or the Korean Index in order to estimate the pitfalls of 2000 and 2010. I haven't been able to do so, unfortunately, since all websites which offered such data required some form of subscription in order for the user to gain access to it.

Contents

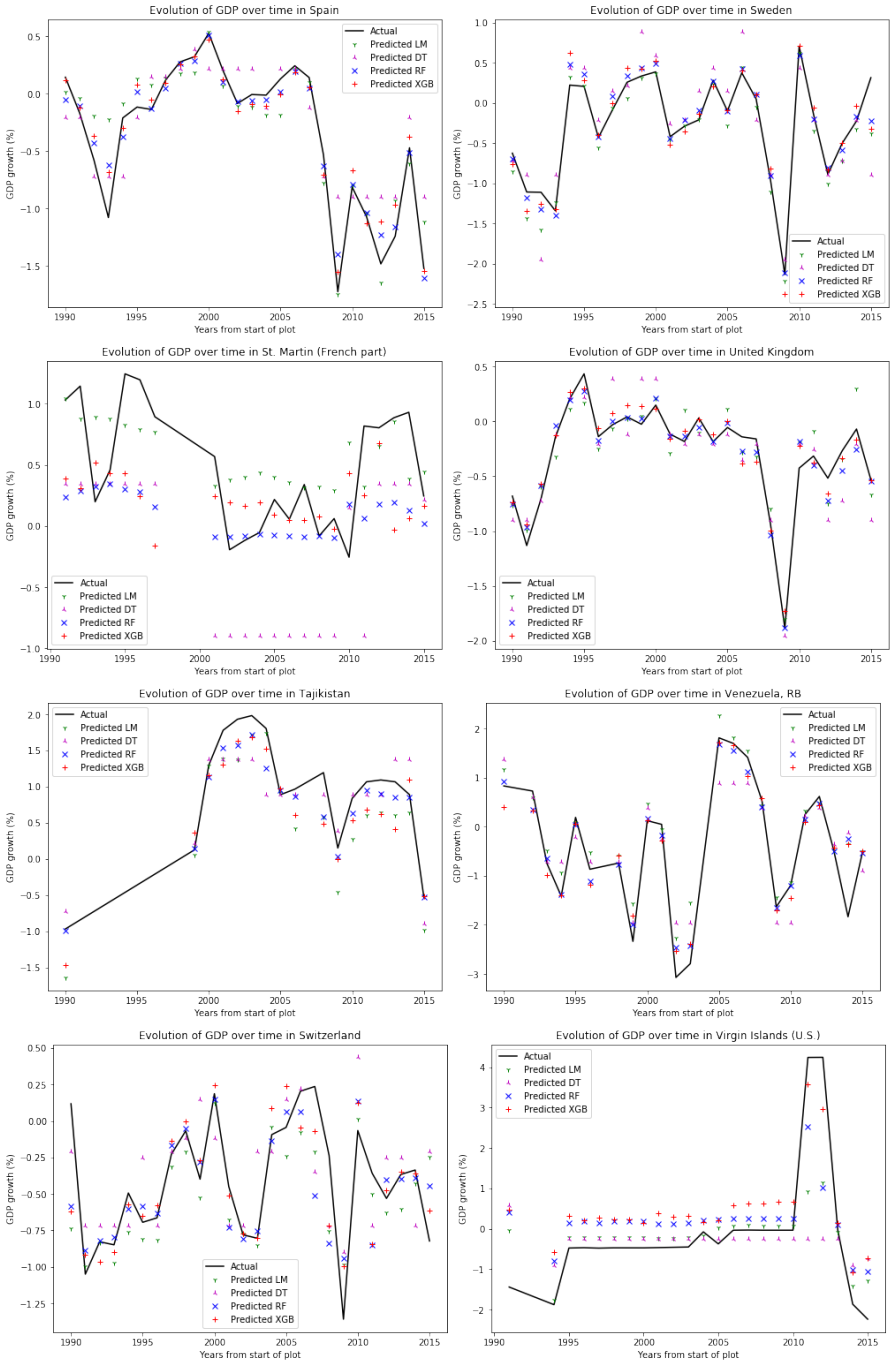

Figure 15: Prediction plots for several European and non-European countries in respect to the features selected by the Random Forest model.

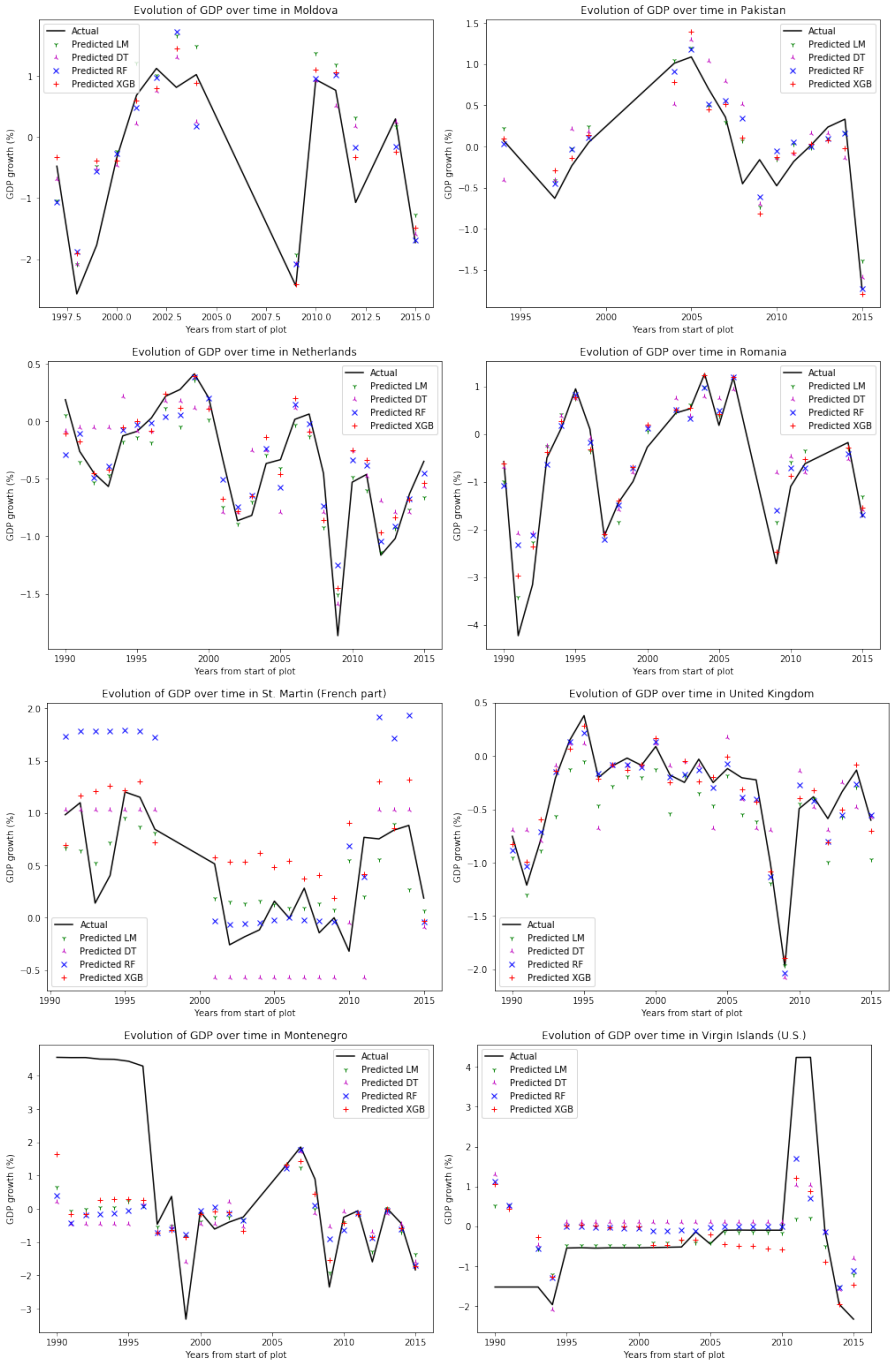

Figure 16: Prediction plots for several European and non-European countries in respect to the features selected by the Lasso model.