PROJECT 1 EDA

Avram Mihai, Ghadamiyan Lida, Pătrașcu Valentin (Class 407 AI)

Neferu Ana-Maria (Class 405 BDTS)

Solar Radiation Prediction

The main goal of this project is predicting solar radiation level given the meteorological statistics of HI-SEAS weather station on a period of four months (09 - 12) in 2016.

Solar radiation represents the main source of energy and heat on Earth, affecting all the living creatures of the planet. It is essential due to the fact that it is the basis of photovoltaic and thermal solar systems. Therefore, the importance of being able to predict this variable plays an essential role in the humankind development, as it is also a renewable source.

1 Data analysis

It is important to carefully analyse the data, both for the preprocessing step, and for the correct evaluation of the project. In our case, we should firstly acknowledge the purpose of predicting the solar radiation, along with the ways that it can be used. After a little round of research we found out that the main use of solar radiation is turning it into energy. So, we would like to create models that generally predict the level of solar radiation, as the process of capturing it implies gadgets that can not be moved easily regarding the level of solar radiation in that day.

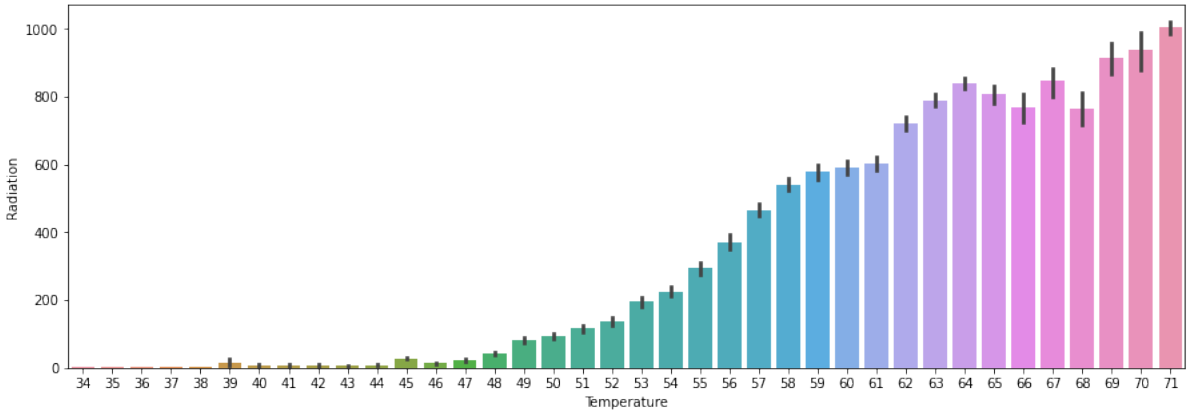
Our data set consists in 32686 unique observations and 11 columns, each of them representing different features as it follows:

1. The UNIX time_t date (seconds since Jan 1, 1970)
2. The date in yyyy-mm-dd format - date when data was collected
3. The local time of day in hh:mm:ss 24-hour format -time when data was collected
4. Solar radiation: $watts$ / $meter^2$ - the one we have to predict
5. Temperature: Fahrenheit degrees - it may be the most important feature since it is a direct consequence of the solar radiation
6. Barometric pressure: Hg- establishes the dependence between atmospheric pressure and altitude

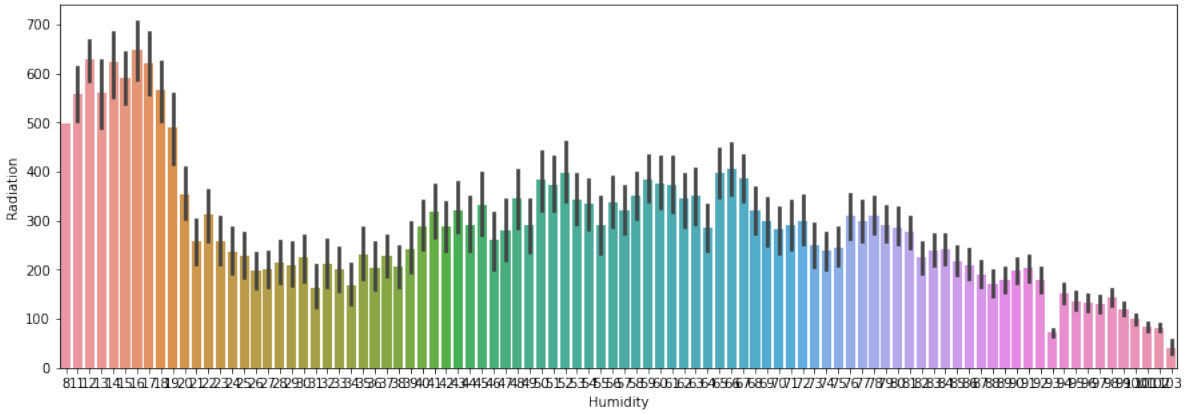
7. Humidity: percent
8. Wind direction: degrees
9. Wind speed: miles per hour
10. Sunrise and sunset: Hawaii time - possible to calculate how many hours of solar radiation there are per day

Intuitively, we believe that the temperature is high when the level of radiation is increased. To check if we have a valid point, we plotted the comparison between the two attributes.

Conform to the plot below, the temperature is directly proportional with the radiation, so it is an important feature, as it strongly influences the level of solar radiation.



In the case of humidity, our intuition says that it blocks the solar radiation. We plot the comparison as we did earlier to verify our instinct. As the humidity level goes down, the radiation tend to grow, so these two attributes seem to be inverse proportional.



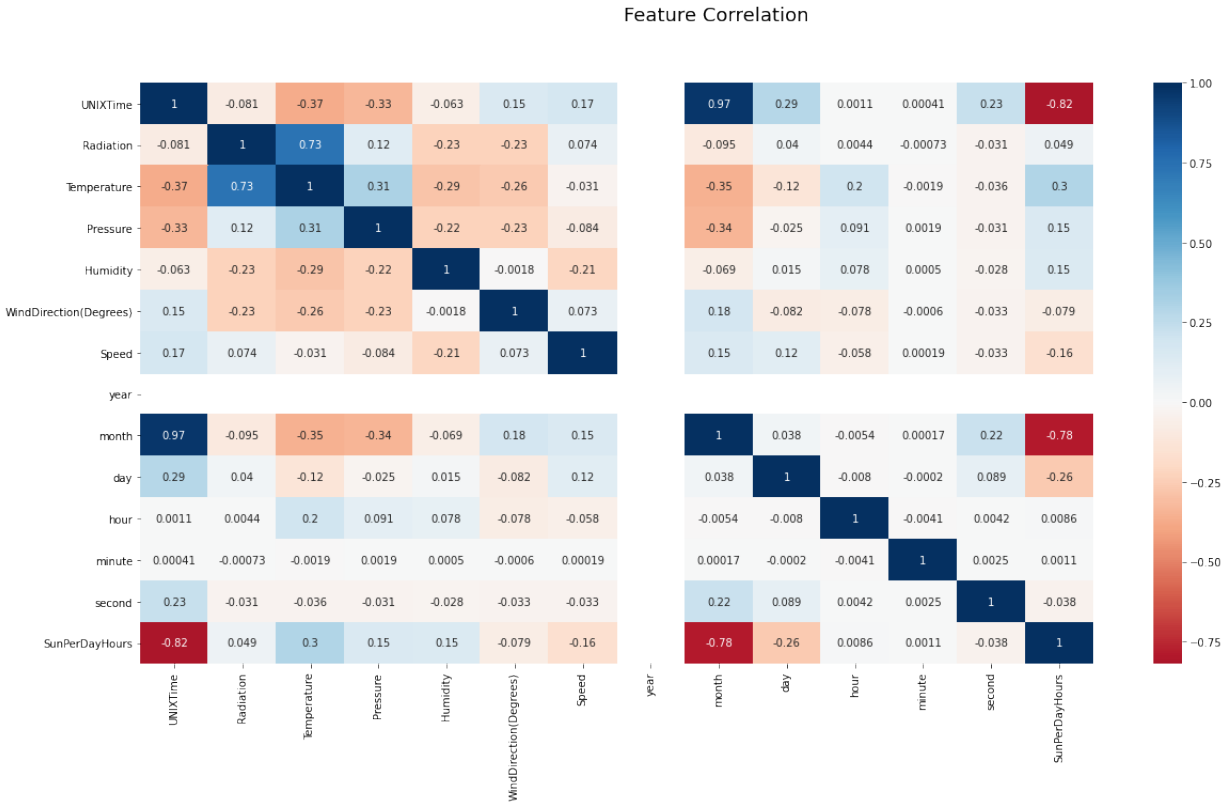
2 Data preprocessing

The first step in the data preprocessing is a good understanding of the data, so we could be able to know how we want it to look and how to manipulate it without losing its properties.

We need the data to be all numeric, so the date and time format features should be transformed in a way that can be represented as int or float. We split Data column into Year, Month, Day and Time into Hour, Minute, Second using DateTimeIndex from panda. We also calculated the exact time gap between TimeSunSet and TimeSunRise and added SunPerDay column with this result. From this column we extracted the hour and we kept it as a feature.

2.1 Correlated Features

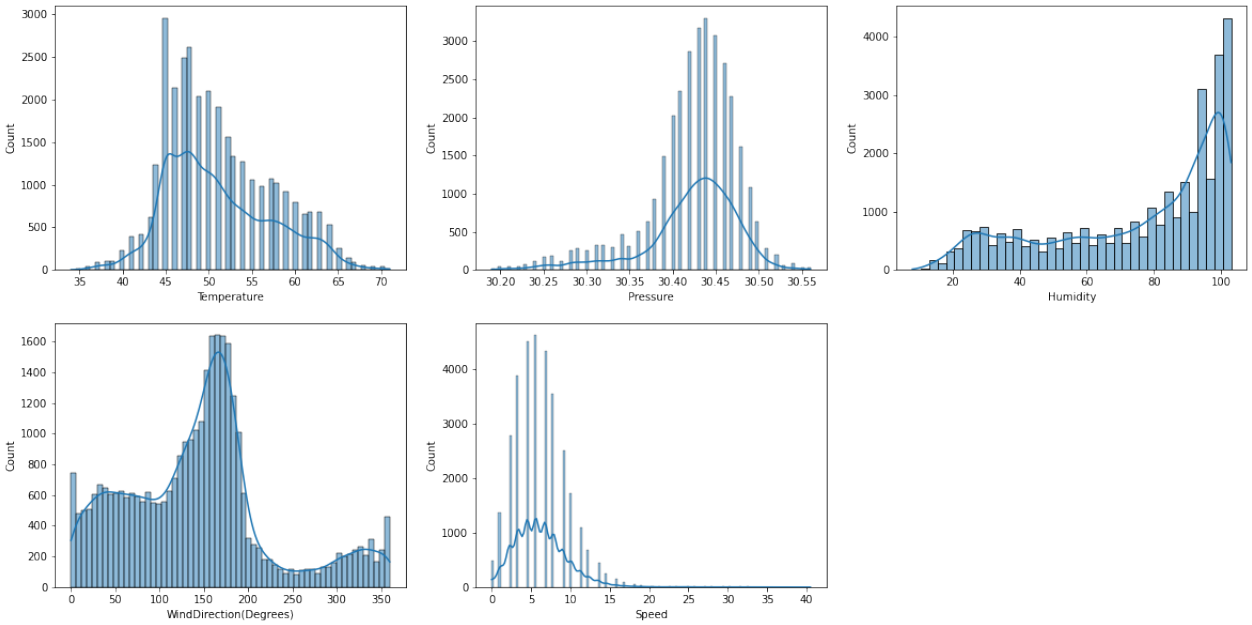
We eliminated the positive correlated features after we analyzed the heat map. Consequently, we dropped the UNIXTime, as we have the exact date when the data was collected and Year, as the data is from the same year, respectively 2016 from our data set.



2.2 Outliers Removal

Removing the outliers can slightly increase the performance of the model. In our case, the data does not present a lot of such formations, as we can see in the following figures that represents our data distribution.

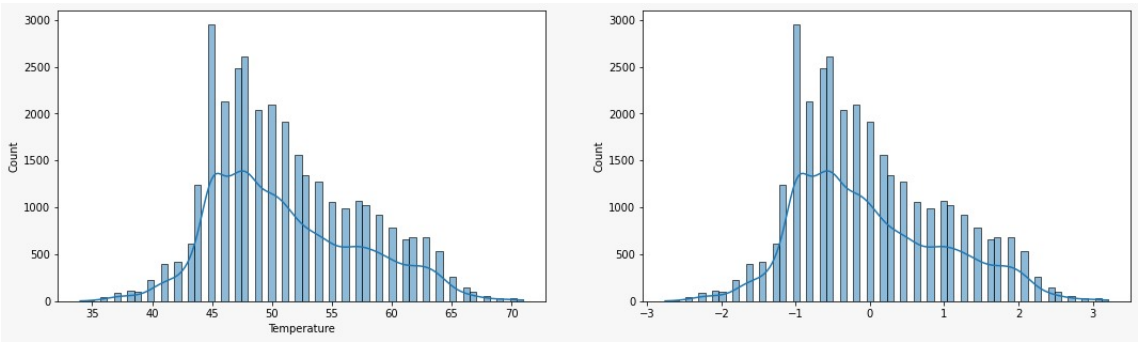
We checked the existence of outliers comparing the Z-score with a fixed threshold. We obtain the Z-score for a variable X by subtracting the mean and dividing by the square root of variance over the the feature dimension. In this way we removed all the existing outliers.



2.3 Data Normalization

Our data needs normalization, because the features have different range of values. The purpose of normalization is to bring the features values to a common scale. We perform standard normalization, substracting the mean and dividing by the standard deviation.

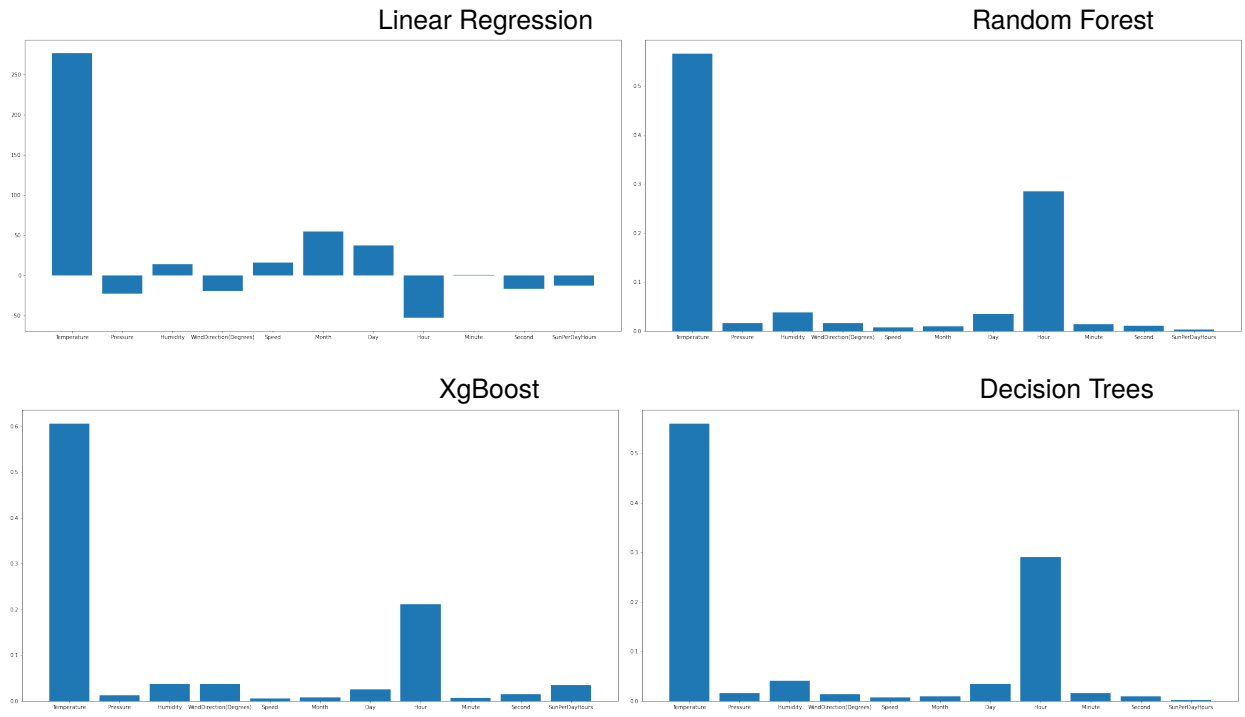
$$normalized\_data = \frac{data - training\_mean}{training\_std}$$



We standardized the features using StandardScaler from scikitlearn and then split the data into test and train in order to be able to evaluate the model.

2.4 Feature importance

We used feature importance to assign a score to each feature based on how important are them for the prediction.



3 Models

3.1 Linear Regression

The first algorithm that we used was Linear Regression. This model attempts to fit a linear equation (Y = AX + B) between an explanatory variable (X) and a dependent variable (Y) to observed data. The best line that fits the data has the smallest distance between the all data points and line itself.

3.2 RandomForest

Random Forest is the second supervised learning algorithm that we used. The model selects a random subset of features to build and merge together a forest of multiple decision trees for getting the most accurate and balanced prediction. The algorithm searches for the best feature through a random subset of features when it divides a node. Only a random batch of features is analyzed by the algorithm while splitting a node. By adding random thresholds for features you can form trees more random. For this project we implemented the Random Forest algorithm using the sklearn library. The following hyperparameters are used for obtaining the proper prediction.

n_estimators = number of trees in the forest

max_features = max number of features considered for splitting a node

max_depth = max depth of the tree

min_samples_split = min number of data points placed in a node before the node is split

min_samples_leaf = min number of data points allowed in a leaf node

bootstrap = method for sampling data points (with or without replacement)

3.3 XgBoost

XGBoost is a algorithm based on decision trees that uses a optimized gradient boosting algorithm to avoid overfitting by parallel processing, tree-pruning, regularization and handling missing values. XGBoost uses two interchangeable loops(inner loop - calculates the features/outer loop - calculates the leaf nodes) for building base learners. For runtime improvement the loop orders can interchanged using a global scan of all instances and sorting using parallel threads. The algorithm stopping criteria depends on the negative loss when the split is happening. On the backward steps XGBoost starts prunning trees for improving computational performance. For this project we implemented the XGBoost algorithm using the xgboost library. The following hyperparameters are used for obtaining the proper prediction.

min_child_weight = min sum of instance weight needed in a child

gamma = min loss reduction to make a further partition on a leaf node

subsample = subsample ratio of the training data

colsample_bytree = subsample ratio of columns when developing each tree

max_depth = max depth of the tree

3.4 DecisionTrees

Decision Tree is a supervised learning algorithm that is used for predicting value of a target variable by using decision rules based on satisfied conditions learned from the training data. In a decision tree, the algorithm starts from the root node (the entire dataset) of the tree. After finding the best attribute in the dateset using Attribute Selection Measure (Information Gain – measures the entropy changes after the dataset segmentation based on an attribute / Gini Index - measures the impurity or purity when a decision tree is created), the algorithm will divide the root node into subsets that have potential of containing the best attributes. The selected attribute will be added in a decision tree node. For this project we implemented the Decision Tree algorithm using the sklearn library. The following hyperparameters are used for obtaining the proper prediction.

max_depth = max depth of the tree

max_features = max number of features considered for splitting a node

random_state = randomness of the estimator by randomly permuting features at each split

min_samples_split = min number of data points placed in a node before the node is split

4 Models Evaluation and Comparison

4.1 Metrics and scoring

For the model evaluation we used Mean Squared Error, Mean Absolute Error, R Squared, Explained Variance Score and Maximum residual error. We studied the first three of them in the course, and the other two are from the scikit-learn documentation regarding regression metrics and scoring .

The Mean Absolute Error represents the mean of the distance between the predictions and the real labels. It is one of the most simple and intuitive form of evaluating a model. This score fails to penalise bigger error as it divides them all by the total number of predictions. Although, it gives us a good overview about the model performance.

$$MAE = \frac{1}{n}\sum_{j=o}^{n-1} |y\_pred_j - y\_true_j|, \text{ where } n = len(y)$$

The Mean Squared Error represents the mean of the quadratic error. It is a good measure if we are looking for a model with a small worst performance, because squaring the error before applying the mean will increase the score much faster for bigger gaps between prediction and true labels.

$$MSE = \frac{1}{n}\sum_{j=o}^{n-1} (y\_pred_j - y\_true_j)^2, \text{ where } n = len(y)$$

The R Squared measure is a statistical score of how good the prediction approximates the real values. A value of one indicates that the model predicted the data perfectly. If the score is bigger than one or smaller than zero, it means that the model behaves worse than a simple horizontal hyperplane, meaning that something was chose wrong or a mistake occurred in the process of implementation.

$$R^2 = \frac{\sum_{j=o}^{n-1}(y\_pred_j - y\_true_j)^2}{\sum_{j=o}^{n-1}(y\_hat_j - y\_true_j)^2}, \text{ where } y\_hat = \frac{1}{n}\sum_{j=o}^{n} y_j \text{ and } n = len(y)$$

The explained variance regression score represents the fraction between the distance of the error from the mean and the distance of the true labels from the mean of the correct results, all subtracted from one. The highest and best value it can take is one and any lower values are worse, so we are looking to have an increased explained variance score.

$$EVS = 1 - \frac{Var(y\_pred - y\_true)}{Var(y\_true)}$$

The Max residual score returns the biggest error between the prediction and the truth. It is an important score, as it shows the worst performance of the model. Like MSE, it give us an overview regarding how badly can our model behave.

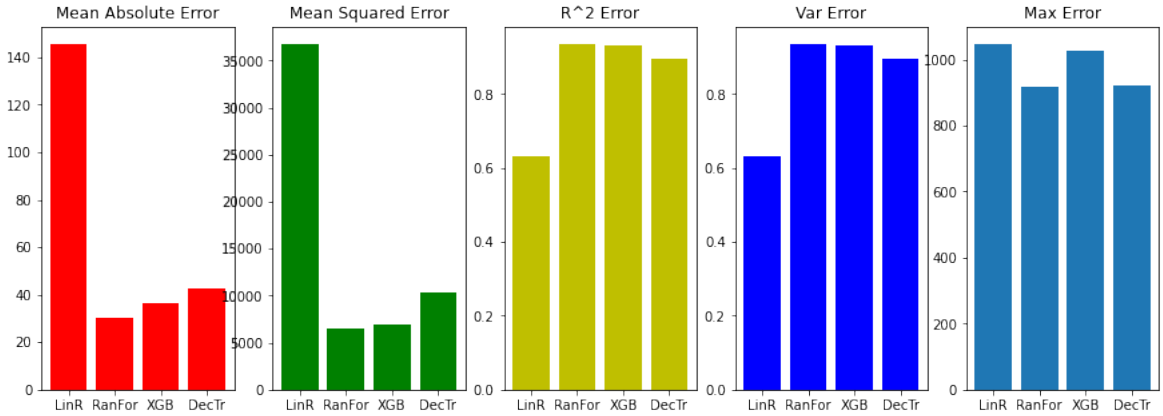
$$max\_err = max_{i \in len(y)}(|y\_true - y\_pred|)$$

4.2 Models comparison

For model comparison we need to choose the model with the smallest MAE, MSE and MaxError and highest R2 and Variance regression score.

Metric	LinearRegression	RandomForest	XGBoost	DecisionTrees
MAE	145.53663438816494	30.319378816151733	36.30675690555069	42.83249842182676
MSE	36765.732838526455	6455.630128860893	6949.370674793824	10346.493150325257
R2	0.6308764850087387	0.9351862535934119	0.9302291581749986	0.8961224590230652
Var	0.6309326107714466	0.9352044899196426	0.9302357563038777	0.8961934722490673
Max	1047.1185829606081	916.6942800000013	1025.7637231445312	922.72375

The table below shows the exact values of the scores for each model. It is quite hard to see the differences in this way, so we computed a histogram for each metric. In this way it is clear how the models behave, and it is easier to choose the most suitable one for our data.



Now we can compare the models for each metric. In the first histogram the Random Forest has the best score, closely followed by XgBoost and DecisionTrees. The worst performance was found using the LinearRegression, which is an expected output, considering the fact that it is a simpler model than the others.

Given the second plot, we can observe that the values are much bigger for MSE than they are for MAE. This means that our models have large errors, even if the medium error is relatively low considering the range value of solar radiation. The succession of the models are the same as for MAE.

In the third and fourth histograms we are looking for the biggest values, that are reached by RandomForest, followed by XgBoost and DecisionTRees. Again, the Linear Regression is the last, but this time the gap between them is not as big as for MAE and MSE.

The least score we are using is the Maximum Error, that gives us the same intuition as MSE, as it clearly shows that the biggest error is very high, about 1000 for each. This time, RandomForest is the best, but closely followed by DecisionTrees. All of the scores are in the same small range (917, 1048), but the two mentioned earlier are lower than 1000, despite the XgBoost and Linear Regression.

In conclusion, the Random Forest Regressor behaved the best, considering all metrics. It is very important to judge regarding multiple metrics, because they all have different meanings. Considering more ways of evaluating the models we have a better understanding of how the prediction look like. In this case, our best model do not behave badly, as the average error is not high, but its worst prediction error are very big. Considering the data, and its purpose, this scenario, of having low MAE and high MSE do not bother us, because the consequences of using this outcomes are for a long term, so the overall results are the one that matters tho most. For example, a consequence of using this model could be the use of solar panels. In this case, large errors for some days would not bother much, as the panels would be fixed for a long time period and the average results are the one that counts.

5 Hyperparameter tunning

We used Grid Search in order to find the best parameters. An important parameter of the Grid Search is the cross validation generator. It randomly split the data in k folds and use one of them for testing and k-1 of them for training until all of them were used for testing. It retain the score while throwing the model. The score is calculated with negative mean squared error.

5.1 Linear Regression

The linear regression module from the sklearn has only 5 posible parameters, but only two of those can influence the performance of the model. Having just 4 possible combinations of parameters (in fact, only three because if we set fit_intercept to False, the normalize parameter will be ignored), the verification of the model performance was manually done.

	MAE	**MSE**	**R2**	**Var**
fit_intercept = True (default)	145.5	36765.7	0.63	0.63
fit_intercept = False (normalize ignored)	224.1	78903.1	0.20	0.63
normalize = True (fit_intercept default)	145.5	36765.7	0.63	0.63

Table 1: Model hyperparameter choices for Linear Regression

As can be seen, when the fit_intercept parameter is set to False, the y-intercept of the linear function is forced to be set in the origin (0,0) and the model will have a constraint, being able to change only the slope of the line (that explains the lower performance results when fit_intercept is set to False)

Another important aspect is that when we set the normalize parameter to True, the performance does not change and that's because we did the normalization of the dataset before the declaration of the LinearRegression model.

5.2 Random Forest

In terms of hyperparameter tunning, for the next three models we used the GridSearchCV module from sklearn.model_selection. To have a non-biased grid search, we used the following parameters for each of the grid searching for every model:

cv=5 scoring='neg_mean_squared_error' verbose=10 n_jobs=-1 Only the first two of them can influence the performance of the grid search and the last two of them influence the feedback from the progress and the last one can influence the time performance (it deals with parallel running on multiple processor).

The first parameter, cv=5 stands for 5-fold cross validation (5 means that we split our training set into 5 different sets and we iteratively fit the model 5 times, each time, using the other 4 folds for training and the evaluation is done on the 5th one). The second one is the method used to evaluate the predictions.

To increase the performance of the Random Forest model, 4 different hyperparameters were selected, with the following values:

'max_depth': [None, 3, 7]

'n_estimators': [100]

'max_features': ['auto', 'sqrt', 'log2']

'max_samples': [None, 0.8]

max_depth was selected to control the complexity of our model and the max_samples was used to control a posible overfit.

We determined empirically that the Random Forest model has pretty good results on our dataset just with the default parameters, without tunning them. So, we also decided to add the standard parameters in the grid search (max_depth:None, n_estimators:100, max_features:auto, max_samples:None)

An extract of 5 combinations from the set of 18 possible combinations of hyperparameters:

max_depth	**max_features**	**max_samples**	**n_estimators**	**mean MSE**
None	auto	None	100	82.11
None	log2	0.8	100	86.83
3	log2	None	100	182.68
7	auto	0.8	100	106.84
7	sqrt	None	100	121.54

Table 2: Grid search results on Random Forest

After the gridsearch finished running we observed that our intuition was true: the default value of the hyperparameters was the one which fits our model best.

5.3 XGBoost

To increase the performance of the XGBoost model, we chose 5 hyperparameters to tune with their correspondent values:

'max_depth': [3, 6, 9]

'min_child_weight': [1, 4, 7]

'gamma': [0, 0.5, 1]

'subsample': [0.8, 1]

'colsample_bytree':[0.8, 1]

- max_depth a high value adds complexity and possible better performances but also overfit.

- min_child_weight the minimum weight required to have a new node spawned in the tree: small value can help generate more complex trees but that are more likely to overfit

- gamma - used for regularization, the higher gamma is, the higher the regularization

- subsample and colsample_bytree used to take subsamples of the rows and columns at each step - prevent overfitting.

An extract of 8 combinations from the set of 540 possible combinations of hyperparameters:

max_depth	**max_child_weight**	**gamma**	**subsample**	**colsample_bytree**	**mean MSE**
3	1	0	0.8	0.8	109.75
6	1	0	0.8	0.8	88.23
9	1	0	0.8	0.8	81.72
9	4	0	0.8	0.8	81.56
9	4	0	1	0.8	82.44
3	7	0.5	1	1	109.45
6	1	0.5	0.8	1	88.22
9	7	1	1	1	82.35

Table 3: Grid search results on XGBoost

After the gridsearch finished running we observed that the best combination of hyperparameters was:

- max_depth=9

- min_child_weight=4

- gamma=0

- subsample=0.8

- colsample_bytree=0.8

5.4 Decision Tree

For the last model implemented we used four parameters to determine the best combination of them:

'max_depth': [None, 3, 5, 10]

'max_features': ['auto', 'sqrt', 'log2']

'random_state': [None, 1, 2, 3, 4]

'min_samples_split': [2, 3, 4]

- the random_state parameter controls the random choises of features and samples

- min_samples_split controls in a way the complexity of the tree

An extract of 8 combinations from the set of 900 possible combinations of hyperparameters:

max_depth	**max_features**	**min_samples_split**	**random_state**	**mean MSE**
None	auto	4	None	111.23
3	auto	3	1	170.62
3	sqrt	4	3	213.34
5	sqrt	4	1	211.87
5	log2	3	4	167.12
10	auto	3	3	101.74
10	log2	2	3	140.57
10	log2	3	4	118.73

Table 4: Grid search results on Decision Tree

6 Further improvements discussed at the presentation

6.1 Prediction without normalization

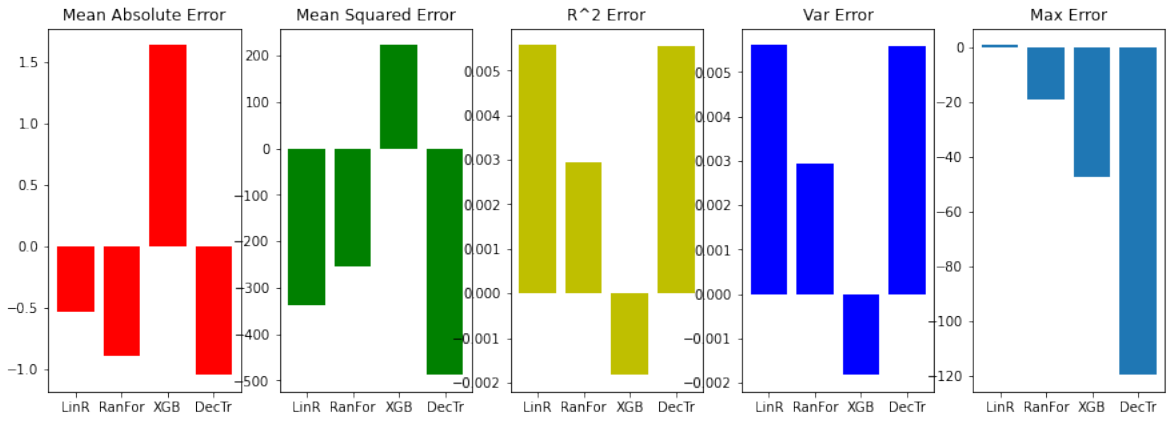
If we get over the normalization step, the results are approximately the same, receiving the following table:

Metric	LinearRegression	RandomForest	XGBoost	DecisionTrees
MAE	145.53663438816486	30.260374034873056	36.45372562953167	42.841940617746474
MSE	36765.7328385265	6494.071158381085	7011.396019976234	10347.625717452076
R2	0.6308764850087383	0.9348003103021767	0.9296064312044026	0.8961110881859694
Var	0.6309326107714461	0.9348176119897434	0.9296161064797657	0.896182079743132
Max	1047.1185829606309	881.2118000000008	1033.2645776367187	922.72375

We computed the following table, containing the rounded differences between the results with and without normalization.

Diference	LinearRegression	RandomForest	XGBoost	DecisionTrees
MAEwn-MAE	-0.53	-0.89	1.63	-1.05
MSEwn-MSE	-336.60	-252.68	222.80	-488.61
R2wn-R2	0.005	0.002	-0.001	0.005
Varwn-Var	0.0056	0.0029	-0.00182	0.0055
Maxwn-Max	0.666	-19.027	-47.246	-119.883

An easier way of watching the differences:



We are looking for negative values for MAE, MSE and MaxError and higher values for R Squarred and Explained Variance Score. We can see that for Linear Regression, Random Forest and Decision-Trees are slightly better.

6.2 Including prunning parameter in GridSearch for Decision Trees

We ran another GridSearch on the Decision Trees, setting the prunning parameter. The output is slightly better, with an improvement of approximately 0.02.

Metric	Old Parameters	New Parameters
MAE	42.841940617746474	42.55817503102817
MSE	10347.625717452076	10098.317452316605
R2	0.8961110881859694	0.8986141130419502
Var	0.896182079743132	0.8986633719110002
Max	922.72375	922.72375

6.3 XbGoost

We performed another GridSearch on XgBoost, adding the learning rate and the number of estimators. The prediction is better now.

Metric	Old Parameters	New Parameters
MAE	36.45372562953167	34.82895552785271
MSE	7011.396019976234	6696.24227124527
R2	0.9296064312044026	0.9327705367590278
Var	0.9296161064797657	0.9327796643591775
Max	1033.2645776367187	1012.6107690429687

7 Conclusion

The solar radiation sustains all the living forms on our planet and determines the climate by providing both light and heat.

We should exploit this form of energy, because it is renewable. To do so, it is important to know where it would be reasonable to install solar panels. We compared four regressors on our data and saw that RandomForest made the best prediction, having the lowest Mean Absolute Error and MaximumError. Given this data, the big values given by the Mean Squared Error do not bother, as these prediction counts for big term decision, so the general results, like Mean Absolute Error are the one that counts the most.

The overall conclusion is that the RandomForest is the best model, the XgBoost followed by DecisionTrees and Linear Regression.