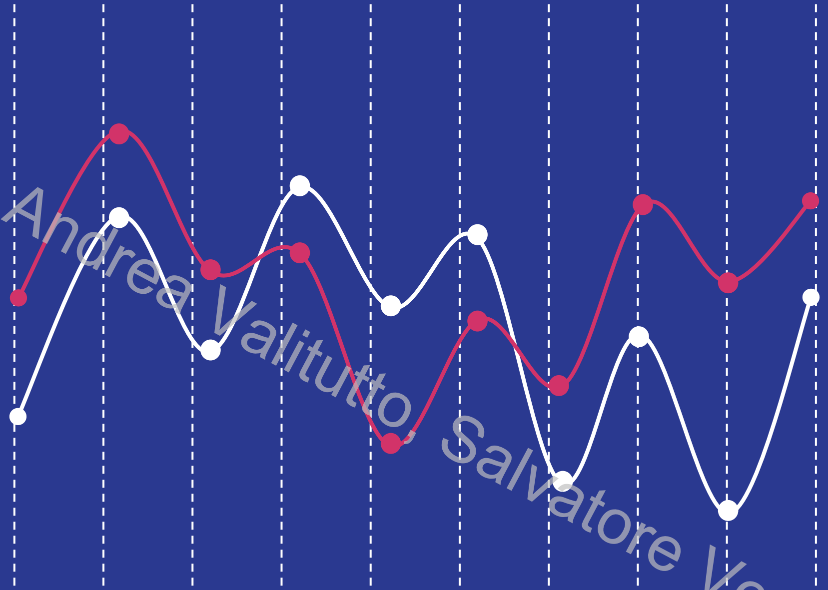
# Statistical Data Analysis

## Project Review

Giovanni Ammendola, Vincenzo Petrone, Andrea Valitutto, Salvatore Ventre



# Team

- Ammendola Giovanni
- Petrone Vincenzo
- Valitutto Andrea
- Ventre Salvatore

# Project

Analyze the same dataset:

- **Classification:**
  - Using Python, Google Colab, Apache Spark & SciKit Learn
- **Regression:**
  - Using R with RStudio, SQLite

# Dataset

The dataset contains user reviews on international airlines posted on skytrax
(The data are available on Kaggle and GitHub):

- **airline:** the name of the airline company
- **traveller_type:** type of traveller (e.g. Business, Family...)
- **class:** cabin class (e.g. Economy, Premium)
- **overall:** the final score assigned to the flight
- **seat_comfort:** score assigned to the comfort
- **cabin_service:** score assigned to the cabin service
- **food_beverage:** score assigned to the food quality
- **entertainment:** score assigned to the entertainment quality
- **ground_service:** score assigned to the ground service
- **wifi_connectivity:** score assigned to the wifi
- **value_for_money:** quality-price ratio
- **recommended:** if the flight is recommended (yes/no)

Final dimensions of dataset: 79.576 rows and 12 variables

# Pre-Processing

This phase was important because two different datasets have been prepared for merge operation:

- **Renaming columns:** renamed every columns of second dataset with the names of the first (except “cabin” and “cabin_flown” in “class”)
- **Change of date format:** change date format in “date_review” column of the first dataset, from day-month-year (e.g. 8th May 2019) into year-month-day (e.g. 2019-05-08)
- **Column removal:** removal of insignificant columns to search equal reviews: “date_flown” (in the first dataset) and “link”, “title” and “author country” (in the second dataset).

# Pre-Processing

- **Editing columns:** in che column “recommended”, the strings “yes/no” were replaced with number values 1 and 0
- **Merging datasets:** the two datasets were merged in one final dataset, in which there are dirty or coarse data
- **Replacement of missing data:** after replacing “0” in “NA” values, missing data were replaced with a predicted approach: PMM (Predictive Mean Matching)
- **Removal of duplicate data:** some rows were present in both the datasets but represented with slightly different values

Final dimensions of the dataset after the preprocessing was of 79.576 rows and 17 columns. The analysis of data caused the removal of five columns: **author**, **review_data**, **customer_review**, **aircraft** and **route**, which are irrelevant for the purpose of the project.

# Python: Classification

- **Main scope:** classify a flight as recommended (class 1) or not (class 0) using the rates of the other aspects of the flight.

- Performed using:
  - Naïve-Bayes Classifier
  - K-Nearest Neighbors
    - Also implemented with Apache Sparks
  - Logistic Regression
    - Also implemented with Apache Sparks
  - Naïve Kernel
    - Implemented only with Apache Sparks

- Synthetic datasets have been used to get a first general idea

# Python: Basic Analysis

- Before applying the actual classification methods, we analyzed some basic statistical properties of the dataset, such as:
- The balancing of the data between classes:
    - 50.41% of data are recommended
    - 49.59% of data are not recommended
- Mean and variance of the features for both recommended and not recommended flights.
    - Example:

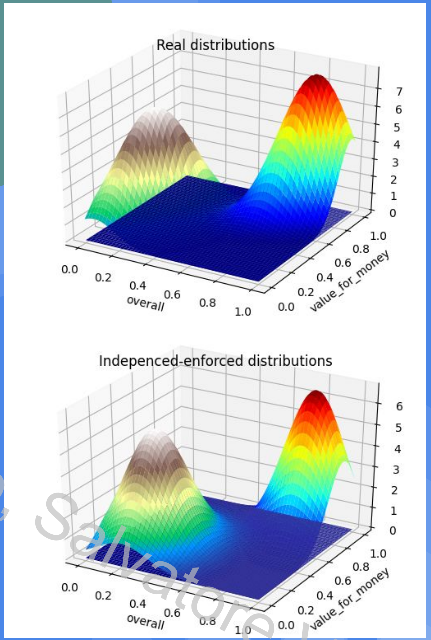
```
For recommeded airlines, the overall mean is 0.8334 with variance 0.0233
For not recommeded airlines, the overall mean is 0.2150 with variance 0.0231

For recommeded airlines, the value_for_money mean is 0.8618 with variance 0.0237
For not recommeded airlines, the value_for_money mean is 0.3431 with variance 0.0368
```

# Python: Synthetic Dataset

- Derived from a normal multivariate with mean and covariance of the real dataset with only two features

- Useful to:
  - get an estimate of the real dataset distribution
  - get an approximation of the independence among features
  - get continuous values instead of discrete ones

- Here are two plots of the synthetic dataset:
  - the top one is the distribution derived from the dataset
  - the covariances of the bottom one have been set to zero
  - the two distributions are not so different, so we can assume that the independence hypothesis holds



# Python: Naïve-Bayes Classifier

- Libraries available on sklearn
- The results are quite high as the independence hypothesis seems to hold
- It comes out that the two most relevant features are **overall** and **value_for_money**

```
Performances...

Confusion matrix:
[[11829   306]
 [  216 11522]]


Classifier performance metrics:
              precision    recall  f1-score   support

         0.0       0.98      0.97      0.98     12135
         1.0       0.97      0.98      0.98     11738

    accuracy                           0.98     23873
   macro avg       0.98      0.98      0.98     23873
weighted avg       0.98      0.98      0.98     23873

final performance:  97.81%
```

```
Performances...

Confusion matrix:
[[11455   541]
 [  499 11378]]


Classifier performance metrics:
              precision    recall  f1-score   support

           0       0.96      0.95      0.96     11996
           1       0.95      0.96      0.96     11877

    accuracy                           0.96     23873
   macro avg       0.96      0.96      0.96     23873
weighted avg       0.96      0.96      0.96     23873

final performance:  95.64%
```

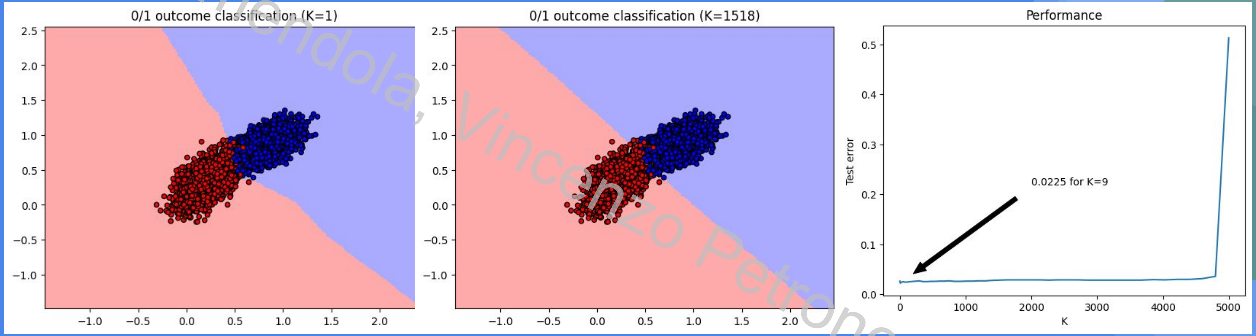
```
Performances...

Confusion matrix:
[[11521   562]
 [  575 11215]]


Classifier performance metrics:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     12083
           1       0.95      0.95      0.95     11790

    accuracy                           0.95     23873
   macro avg       0.95      0.95      0.95     23873
weighted avg       0.95      0.95      0.95     23873

final performance:  95.24%
```

| overall | seat_comfort | cabin_service | food_bev | entertainment | ground_service | wifi_connectivity | value_for_money |
|---|---|---|---|---|---|---|---|
| 95.33% | 82.30% | 86.08% | 81.62% | 77.03% | 84.48% | 73.90% | 91.01% |

# Python: K-Nearest Neighbors

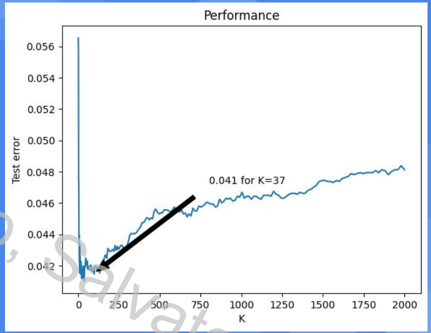
- Implemented with sklearn library and manually with Spark
- We defined the following functions, used with Spark:
  - **`euclidean_distance`**: to retrieve the distance between two points
  - **`K_nearest_neighbors`**: to retrieve the `K` nearest neighbors with respect to a sample.
- The results from sklearn and Sparks are the same on the synthetic dataset, so we used only sklearn on the real dataset.
- KNN has been implemented with
  - **value_for_money** and **overall**, from synthetic dataset
  - **value_for_money** and **overall**, from real dataset
  - all features one by one
  - all features together
- The performances are almost the same as Naïve-Bayes Classifier

# Python: K-Nearest Neighbors




Example and result for the KNN on two features. For greater values of K, the test error increases and the decision regions are split by a straight line (5000 samples)

Performances for different K on the real dataset with all features

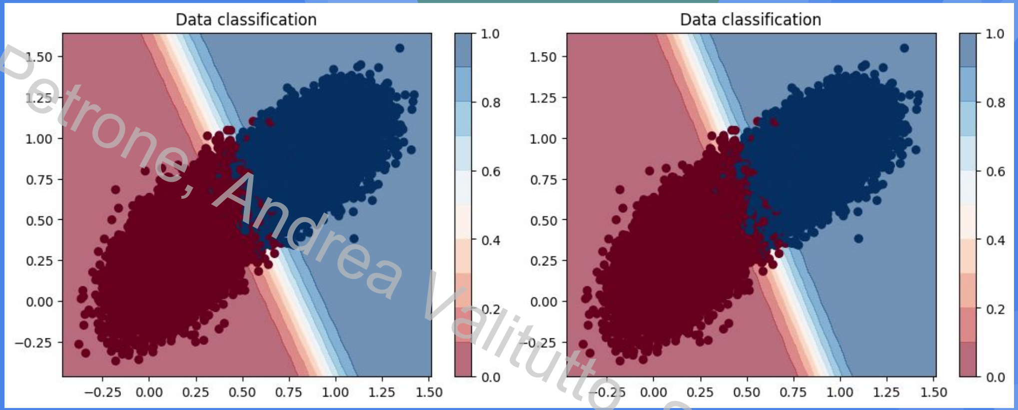



| | overall | seat_comfort | cabin_service | food_bev | entertainment | ground_service | wifi_connectivity | value_for_money |
|---|---|---|---|---|---|---|---|---|
| K | 108 | 149 | 311 | 1241 | 1306 | 640 | 1339 | 99 |
| MSE | 4.17% | 17.30% | 13.30% | 17.20% | 21.80% | 15.10% | 24.40% | 8.56% |
| Score | 95.46% | 81.90% | 85.85% | 81.80% | 77.13% | 84.09% | 74.69% | 90.75% |

KNN Performances for different Ks and different features on the real dataset

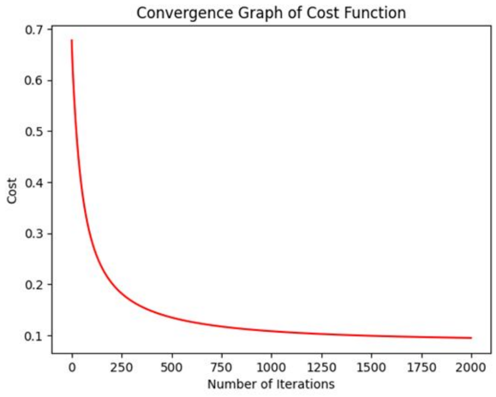
# Python: Logistic Regression

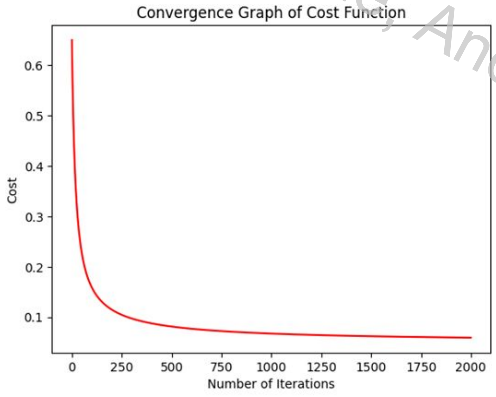
- Implemented with sklearn library and manually with Spark
- We defined the following functions:
  - sigmoid
  - local gradient
  - cost function
  - gradient descent algorithm
  - predict function
- The results with the two implementations are very similar

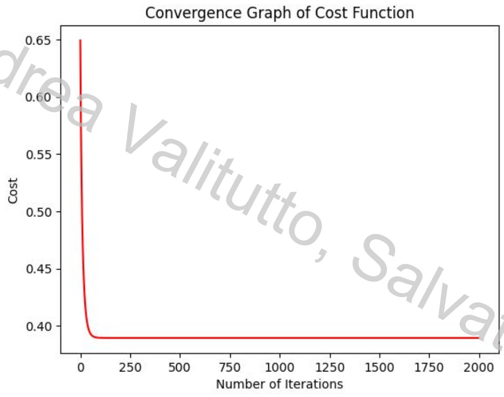


# Python: Logistic Regression Cost Function

- This graphs show the convergence of the cost function with respect to the learning rate parameter
- As it increases, it converges faster, but it can get stuck at a different minimum value



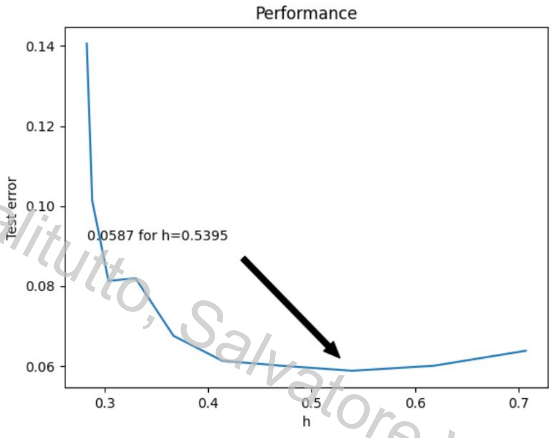
# Python: Naïve Kernel

- Implemented only manually with Spark
- We defined the following functions:
  - **`euclidean_distance`**: to retrieve the distance between two points
  - **`neighbors_in_h`**: to retrieve all the points in a given radius, h, from a given sample

These are the final performances and the test error for different `h` values

Final performance: 88.00%

| | overall | seat_comfort | cabin_service | food_bev | entertainment | ground_service | wifi_connectivity | value_for_money |
|---|---|---|---|---|---|---|---|---|
| K | 0.3666 | 0.2828 | 0.3666 | 0.3666 | 0.2881 | 0.2881 | 0.3666 | 0.2828 |
| MSE | 5.50% | 17.25% | 13.00% | 14.50% | 18.50% | 13.75% | 23.00% | 9.75% |
| Score | 93.17% | 80.50% | 84.33% | 80.67% | 75.17% | 83.83% | 78.17% | 88.00% |



# R: Regression

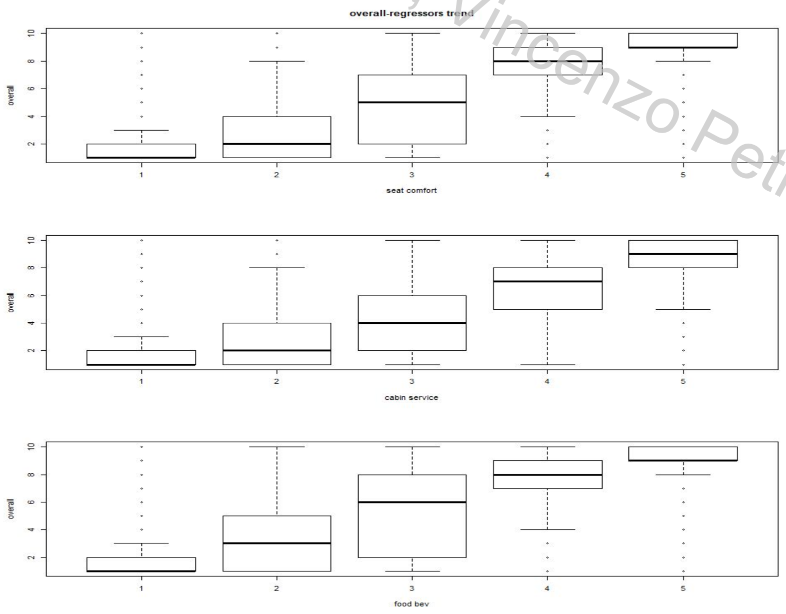
**Main goal:** predict “overall” wrt regressors present in the dataset (recommended, seat_comfort, cabin_service, food_bev, entertainment, ground_service, wifi_connectivity, value_for_money).

**Regression Methods used:**

- Multiple Regression
- Resampling Methods
  - K-Fold Cross Validation
  - Bootstrap
- Best Subset Selection
- Stepwise Selection
  - Forward Stepwise Selection
  - Backward Stepwise Selection
- Regression with Regularization
  - Ridge and Lasso
- PCR and PLS

# R: Multiple Regression

Check the trend between dependent variable (overall) and regressors:




As we can see in the figure, the trend between "overall" and its regressors is not linear. So probably a linear model does not interpret the data in the best possible way.

# R: Multiple Regression

After comparing all the models by analyzing the "summary(model)" and "anova(model1,model2)" functions, the result led us to say the model that best interprets our data is a model with order 4 transformation of regressors:

```
Analysis of Variance Table

Model 1: overall ~ recommended + poly(seat_comfort, 3) + poly(cabin_service,
    3) + poly(food_bev, 3) + poly(entertainment, 3) + poly(ground_service,
    3) + poly(wifi_connectivity, 3) + poly(value_for_money, 3)
Model 2: overall ~ recommended + poly(seat_comfort, 4) + poly(cabin_service,
    4) + poly(food_bev, 4) + poly(entertainment, 4) + poly(ground_service,
    4) + poly(wifi_connectivity, 4) + poly(value_for_money, 4)
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1  79553 87754
2  79546 87640  7    114.27 14.817 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is the comparison, with anova function, between order 3 transformation (model 1) and order 4 transformation (model 2)

# R: Multiple Regression

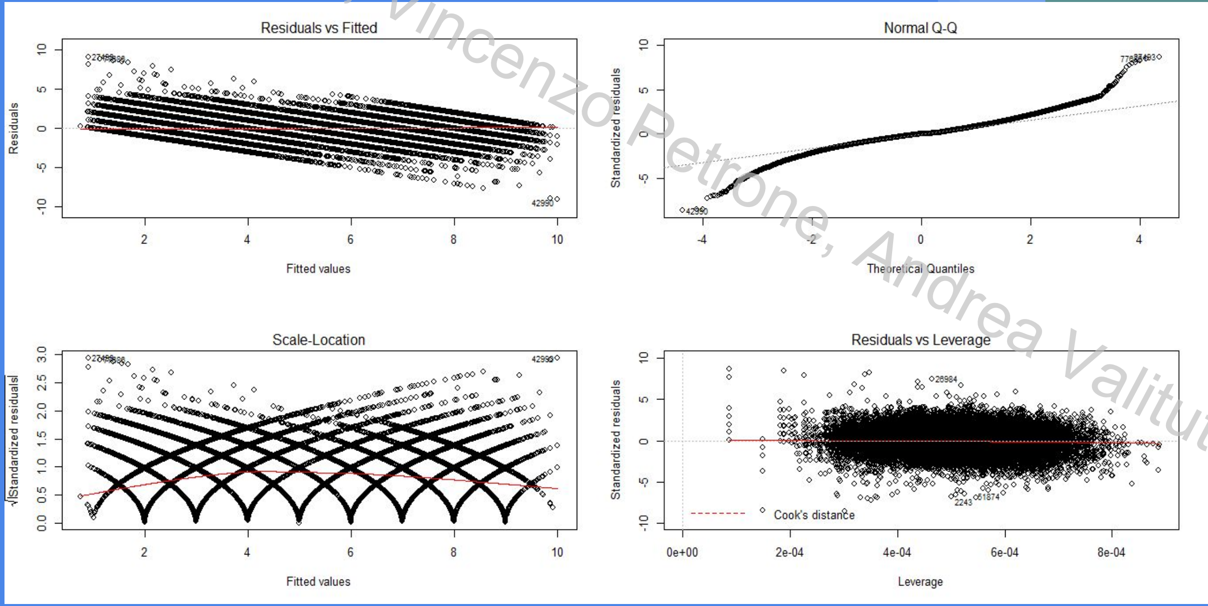
This is the output of "summary(model)" function:

```
Coefficients:
                               Estimate Std. Error t value Pr(>|t|)    
(Intercept)                    4.039990   0.008855 456.260  < 2e-16 ***
recommended                    2.382418   0.016202 147.049  < 2e-16 ***
poly(seat_comfort, 4)1        80.197151   1.714849  46.766  < 2e-16 ***
poly(seat_comfort, 4)2        10.467691   1.172241   8.930  < 2e-16 ***
poly(seat_comfort, 4)3        -0.903590   1.087648  -0.831  0.40610    
poly(seat_comfort, 4)4        -6.203441   1.056802  -5.870 4.37e-09 ***
poly(cabin_service, 4)1      121.074678   1.853455  65.324  < 2e-16 ***
poly(cabin_service, 4)2       22.933709   1.187773  19.308  < 2e-16 ***
poly(cabin_service, 4)3        6.930331   1.086816   6.377 1.82e-10 ***
poly(cabin_service, 4)4       -4.993480   1.056648  -4.726 2.30e-06 ***
poly(food_bev, 4)1            69.995053   1.790512  39.092  < 2e-16 ***
poly(food_bev, 4)2            14.019276   1.188120  11.800  < 2e-16 ***
poly(food_bev, 4)3            -0.281704   1.085940  -0.259  0.79532    
poly(food_bev, 4)4            -3.311297   1.055235  -3.138  0.00170 ** 
poly(entertainment, 4)1       48.513183   1.619982  29.947  < 2e-16 ***
poly(entertainment, 4)2        3.068035   1.137910   2.696  0.00701 ** 
poly(entertainment, 4)3        2.366611   1.061798   2.229  0.02583 *  
poly(entertainment, 4)4       -2.320536   1.053434  -2.203  0.02761 *  
poly(ground_service, 4)1     167.295629   1.752951  95.437  < 2e-16 ***
poly(ground_service, 4)2      -5.637883   1.133313  -4.975 6.55e-07 ***
poly(ground_service, 4)3       4.181108   1.068695   3.912 9.15e-05 ***
poly(ground_service, 4)4      -4.773979   1.052140  -4.537 5.70e-06 ***
poly(wifi_connectivity, 4)1  -17.621380   1.600336 -11.011  < 2e-16 ***
poly(wifi_connectivity, 4)2    7.814279   1.129980   6.915 4.70e-12 ***
poly(wifi_connectivity, 4)3   -2.034234   1.063157  -1.913  0.05570 .  
poly(wifi_connectivity, 4)4   -0.822053   1.065209  -0.772  0.44028    
poly(value_for_money, 4)1    234.162317   2.277116 102.833  < 2e-16 ***
poly(value_for_money, 4)2     -7.965631   1.172366  -6.794 1.09e-11 ***
poly(value_for_money, 4)3     -8.351883   1.138266  -7.337 2.20e-13 ***
poly(value_for_money, 4)4      0.375848   1.056655   0.356  0.72207    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.05 on 79546 degrees of freedom
Multiple R-squared:  0.9071,	Adjusted R-squared:  0.9071 
F-statistic: 2.678e+04 on 29 and 79546 DF,  p-value: < 2.2e-16
```

**Least significant predictors are**:
seat_comfort,
food_bev,
entertainment and
wifi_connectivity

# R: Multiple Regression

These are some diagnostic graphs produced by the output of “lm()” function:



# R: Validation Set Approach

The dataset is divided in two parts:

- train_set
- test_set

each part has a number of 39.788 samples. The obtained results are:

| Transformation | MSE |
|---|---|
| Linear | 1.111269 |
| Polynomial-2 | 1.095797 |
| Polynomial-3 | 1.094223 |
| Polynomial-4 | 1.093318 |
| Logarithmic | 1.257818 |

# R: Resampling Methods

## K-FOLD CROSS VALIDATION

The dataset is divided in 10 subgroups (folds) with a limited random data.
This number indicates the iterations performed for the training and test of the model.

The values obtained are shown in the following table:

| Transformation | MSE |
|---|---|
| Linear | 1.120662 |
| Polynomial-2 | 1.105010 |
| Polynomial-3 | 1.103584 |
| Polynomial-4 | 1.102515 |

# R: Resampling Methods

## BOOTSTRAP

In the project the bootstrap was used to quantify the discrepancy between the estimate of the standard error of the coefficients.

A small difference in the estimate of the standard error is synonymous of a good interpretability of the data by the chosen model.

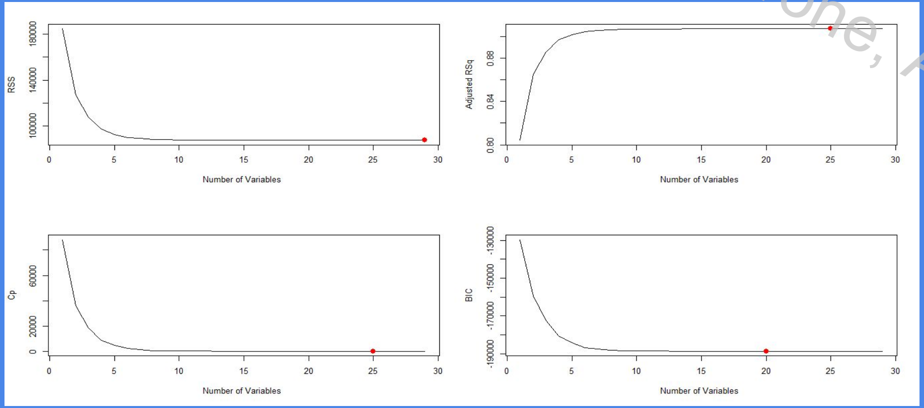
| Coefficients | Standard Error difference |
|---|---|
| Intercepts | 0.0004636287 |
| recommended | -0.006179144 |
| seat_comfort | -0.0006253075 |
| cabin_service | -0.0009786729 |
| food_bev | -0.000399543 |
| entertainment | -0.0005200939 |
| ground_service | -0.001155898 |
| wifi_connectivity | -0.0002902589 |
| value_for_money | -0.001460509 |

# R: Subset Selection

### Best Subset Selection

This approach is the process of selecting a subset of predictors for use in building the model. It was applied to different transformations.

The model selection criteria are: RSS, Adjusted $R^2$, Cp and BIC.




**Polynomials-4**

*RSS:* min at 29 predictors

*Adjusted $R^2$*: max at 25 predictors

*BIC:* min at 20 predictors

*Cp*: min at 25 predictors

# R: Subset Selection

## Final Considerations (1)

To evaluate the performance of the best model, found by these three techniques, the Validation Set Approach was used. The fit has been performed on the training set and then the MSE was estimated on the test set. The best model that provides the least MSE is the one with transformation of order four of the predictors.

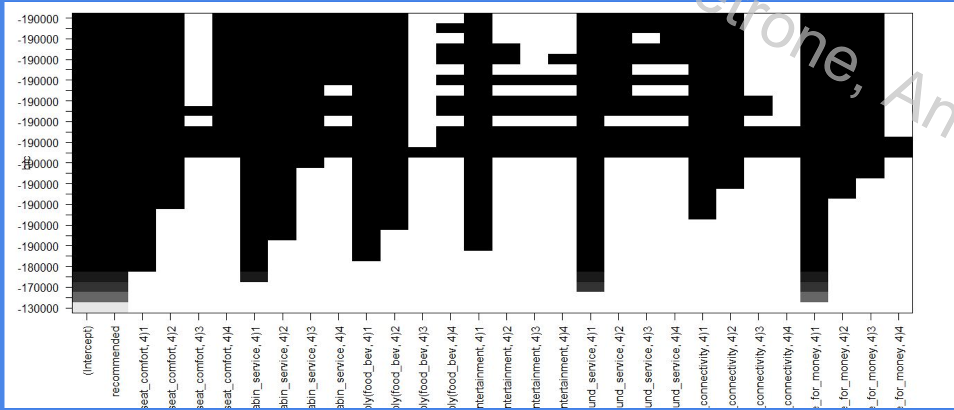
The values obtained are shown in the following table:

| Transformation | Predictors considered | Min MSE |
|---|---|---|
| Linear | 8 | 1.106853 |
| Polynomial-2 | 14 | 1.090422 |
| Polynomial-3 | 22 | 1.089150 |
| Polynomial-4 | 28 | 1.087877 |

# R: Subset Selection

## Final Consideration (2)

Considering the transformation of the fourth order, relating to the BIC parameter, the value of the number of predictors with this parameter is minimal:




The predictors not considered by these subset selection approaches are **seat_comfort**, **food_bev**, **entertainment** and **wifi_connectivity**.

# R: Subset Selection

### Final Considerations (3)

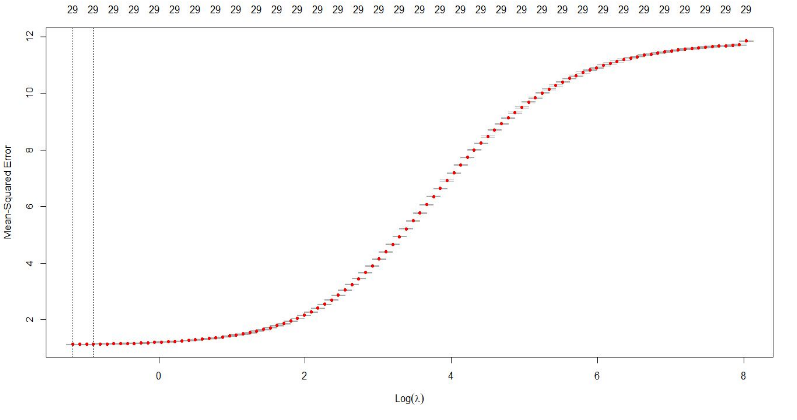
Following the one-standard-error-rule, the suggested model to select is the one that minimizes the BIC, as it is the one that provides the least number of predictors (20).

```
Model 1: overall ~ recommended + poly(seat_comfort, 2) + I(seat_comfort^4) +
    poly(cabin_service, 4) + poly(food_bev, 2) + entertainment +
    poly(ground_service, 4) + poly(wifi_connectivity, 2) + poly(value_for_money,
    3)
Model 2: overall ~ recommended + poly(seat_comfort, 4) + poly(cabin_service,
    4) + poly(food_bev, 4) + poly(entertainment, 4) + poly(ground_service,
    4) + poly(wifi_connectivity, 4) + poly(value_for_money, 4)
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1  79555 87714
2  79546 87640  9    74.282 7.4913 4.922e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

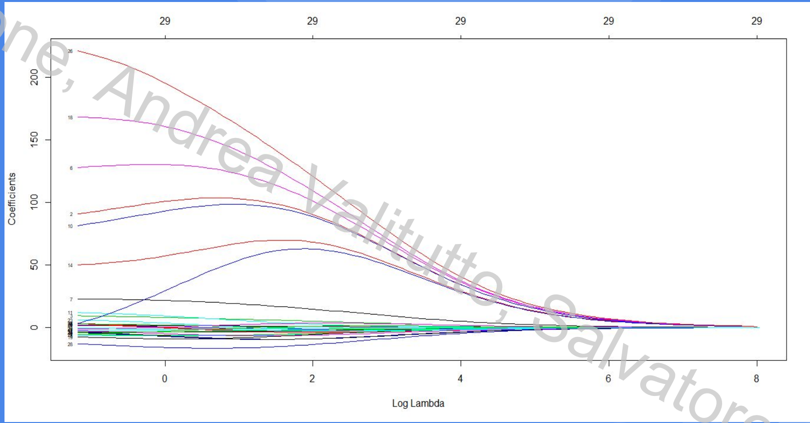
For a number of predictors above 20, the other parameters do not improve significantly; therefore, even if the other parameters suggest selecting a more complex model, for the same performance it is preferable to select the simplest model, in order to prevent overfitting problems.

# R: Regularization: Ridge

To choose $\lambda$, after building a grid of 100 values in a range from $10^{-2}$ to $10^{10}$, is used cross validation to estimate the best $\lambda$. This is the graph of the mse as a function of $\lambda$ **(polynomial-4 transformation)**:




This is the graph of the coefficients as a function of $\lambda$:



# R: Regularization: Ridge

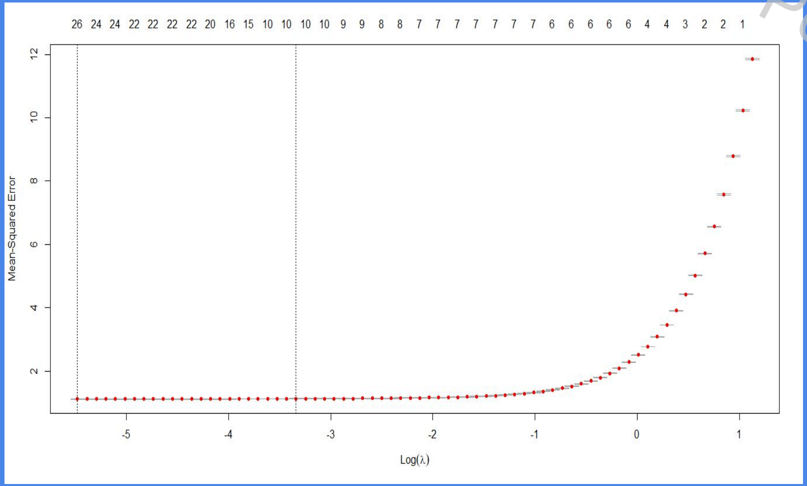
Results using **Ridge regularization**: Ridge does not bring improvements in the model's performance wrt no regularization.

| Transformation | MSE no regularization | MSE regularization |
|---|---|---|
| Linear | 1.120662 | 1.128454 |
| Polynomial-2 | 1.105010 | 1.11337 |
| Polynomial-3 | 1.103584 | 1.112249 |
| Polynomial-4 | 1.102515 | 1.111561 |

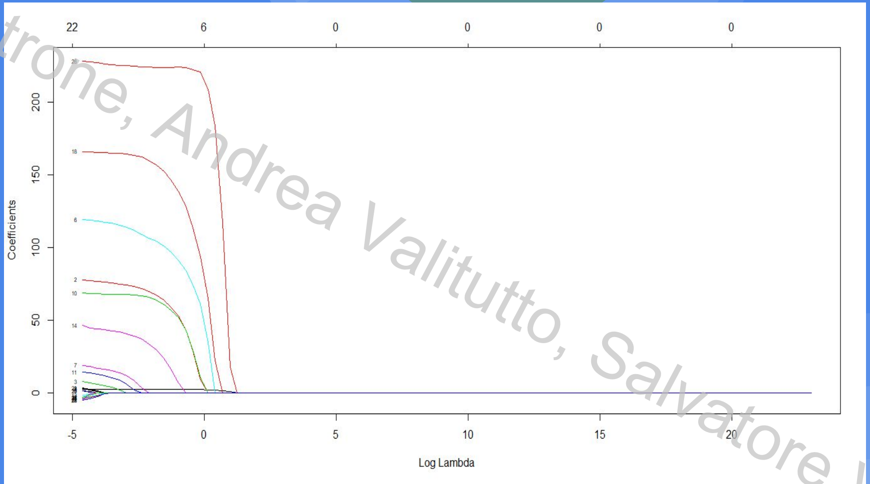
# R: Regularization: LASSO



**This is the graph of the mse as a function of** $\lambda$ (polynomial-4 transformation):




**This is the graph of the coefficients as a function of** $\lambda$**:**



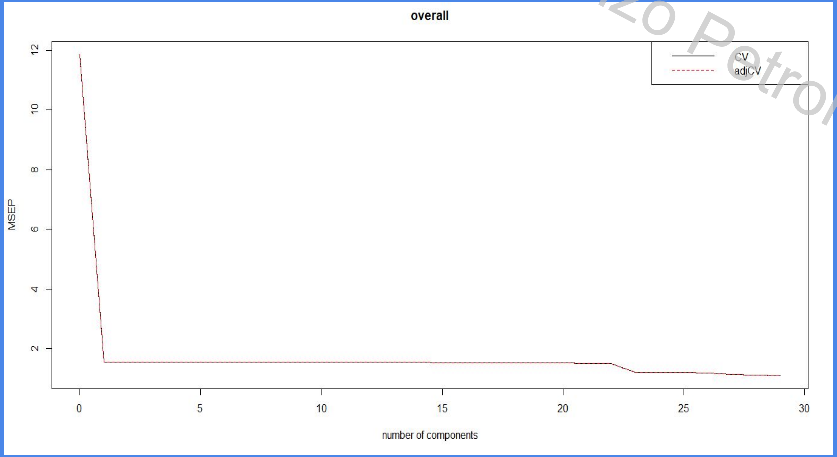
# R: Regularization: LASSO

Results using **LASSO regularization**: LASSO brings improvements in model performance and, unlike ridge, produces much simpler and interpretable models with better performance.

| Transformation | MSE no regularization | MSE regularization | Coefficients equal to zero |
|---|---|---|---|
| Linear | 1.120662 | 1.112508 | 1 |
| Polynomial-2 | 1.105010 | 1.097244 | 0 |
| Polynomial-3 | 1.103584 | 1.096104 | 4 |
| Polynomial-4 | 1.102515 | 1.095203 | 7 |

# R: PCR

The PCR procedure calculates two main results: "Root MSE" and the percentage of variance explained using n components.



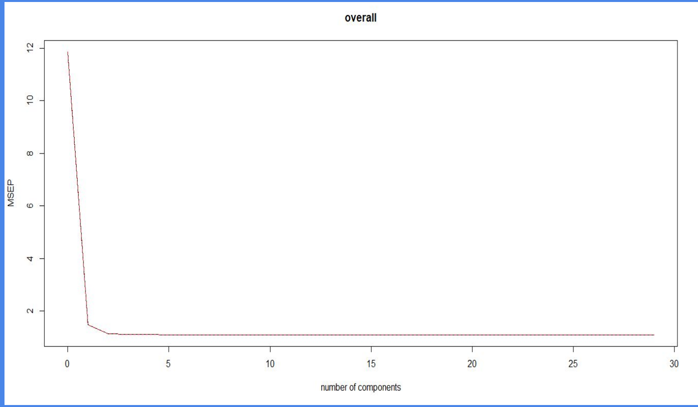

As the graph shows, the minimum MSE is obtained by considering all 29 predictors.

Moreover, using a fourth-order transformation of predictors, the percentage of variance explained on the overall turns out to be 90.71% considering all predictors.

# R: PLS

The **PLS** produces same output of the PCR: Root MSE and the variance explained as the number of components changes.

**MSEP** as a function of number of the components considered in the model **(Polynomial-4 transformation):**




**Components number considered to obtain min MSE wrt all transformation:**

| Transformation | Components Number (min. MSE) |
|---|---|
| Linear | 8 su 8 |
| Polynomial-2 | 14 su 15 |
| Polynomial-3 | 15 su 22 |
| Polynomial-4 | 12 su 29 |

# R: PLS

The subsequent analysis performed included the fit using the components found by the **PLS** on the training set and the evaluation on a test set.

| Transformation | Components Number | min. MSE |
|---|---|---|
| Linear | 7 | 1.111269 |
| Polynomial-2 | 9 | 1.095787 |
| Polynomial-3 | 11 | 1.094217 |
| Polynomial-4 | 10 | 1.093311 |

Finally the analysis of the minimum number of components to have the maximum percentage variance explained on the variable Y.

| Transformation | Components Number | PVE |
|---|---|---|
| Linear | 4 su 8 | 90.55% |
| Polynomial-2 | 5 su 15 | 90.68% |
| Polynomial-3 | 7 su 22 | 90.70% |
| Polynomial-4 | 6 su 29 | 90.71% |

# R: PCR - PLS Conclusion

- The number of predictors considered by the PLS is lower than those considered by the PCR.
- The maximum variance explained by predictors is achieved through fewer variables.
- As for the MSE, its values are in line with those found with Ridge and LASSO.

# Thanks for your Attention

## Statistical Data Analysis