NOVA IMS Information Management School

ML

OBESITY LEVEL

Predicting the Obesity Level using different characteristic of a person

Group 04

Date 20/12/2024

Cyrine Moalla Name 20230003

Darija Avramoska 20230004

Khadija Belhamra 20230007

Victoriya Mokhovikova 20230796

Vladislav Botnev 20230795

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Abstract

Universidade Nova de Lisboa

Obesity is a critical public health issue linked to sedentary lifestyles and poor dietary habits. This study aims to develop a machine learning model to predict obesity levels based on lifestyle, and physical health data.

The pipeline for data preprocessing is: exploration, cleaning, encoding, imputation, feature engineering, outliers, and scaling.

Models: Logistic Regression, kNN classifier, BaggingClassifier with trees, trees, SVC, MLP, Naive Bayes, Random Forest, Gradient Boosting, Stacking**.**

The project implemented and optimized various algorithms, including Random Forest, Gradient Boosting, and Stacking, achieving high predictive accuracy. We tried different combinations of models in Stacking and chose one of them. Results highlighted BMI and physical activity as significant factors, reinforcing their importance in obesity prediction. These findings emphasize the role of machine learning in public health by offering insights to guide targeted interventions and preventive measures.

Results: The best score on Kaggle is 0,9843. In stacking base models should belong to different types of algorithms and the metamodel should be linear.

Discussion: our target variable has a straightforward relationship with BMI, which we verified and tested using Excel formula. This approach gave a good score, showing the importance of this variable in determining the target.

In conclusion, the developed model provides a reliable framework for predicting obesity and informing public health strategies aimed at mitigating risks and promoting healthier lifestyles.

Keywords

Machine Learning; Model; Optimization; Feature Engineering; Prediction; Boosting; Stacking

INTRODUCTION

Obesity has emerged as one of the most pressing public health challenges of the 21st century, affecting individuals across all demographics and geographic regions. Its prevalence has been steadily increasing due to lifestyle factors such as physical inactivity, poor dietary habits, and urbanization. The condition is associated with severe health complications, including cardiovascular diseases, diabetes, and certain cancers, which impose significant burdens on healthcare systems worldwide.

Understanding the factors contributing to obesity and predicting obesity levels are crucial steps toward addressing this epidemic. Machine learning techniques provide a powerful approach to uncover patterns in complex datasets and offer predictive insights. By analyzing behavioral, demographic, and health-related data, machine learning can identify key drivers of obesity and support targeted interventions.

This study aims to develop a predictive Machine learning model capable of accurately estimating an individual's obesity level based on a dataset that captures diverse attributes, including dietary habits, physical activity levels, and familial health history. The primary objective is to explore the relationship between lifestyle factors and obesity levels, providing insights that can inform public health strategies and preventive measures.

Through this project, we seek to address the growing need for data-driven solutions to combat obesity. By leveraging machine learning, this work contributes to a better understanding of obesity's multifaceted nature and provides actionable insights for policymakers and healthcare providers.

Data Exploration

The data exploration is an essential phase of our project for understanding the dataset and identifying different patterns. This step includes analyzing the dataset, distributions, and relationships between features to get usable insights for data preprocessing and modeling.

The dataset consists of 1611 entries with 20 features, including both categorical and numerical variables. The target variable, ***obese_level***, represents different obesity categories such as Insufficient_Weight, Normal_Weight, Overweight_Level_I, Overweight_Level_II, Obesity_Type_I, Obesity_Type_II, Obesity_Type_III. These features provide a detailed view of individual health and lifestyle habits.

Numerical features : *Age, height, weight, meals_perday* and *sibling*

From descriptive statistics :

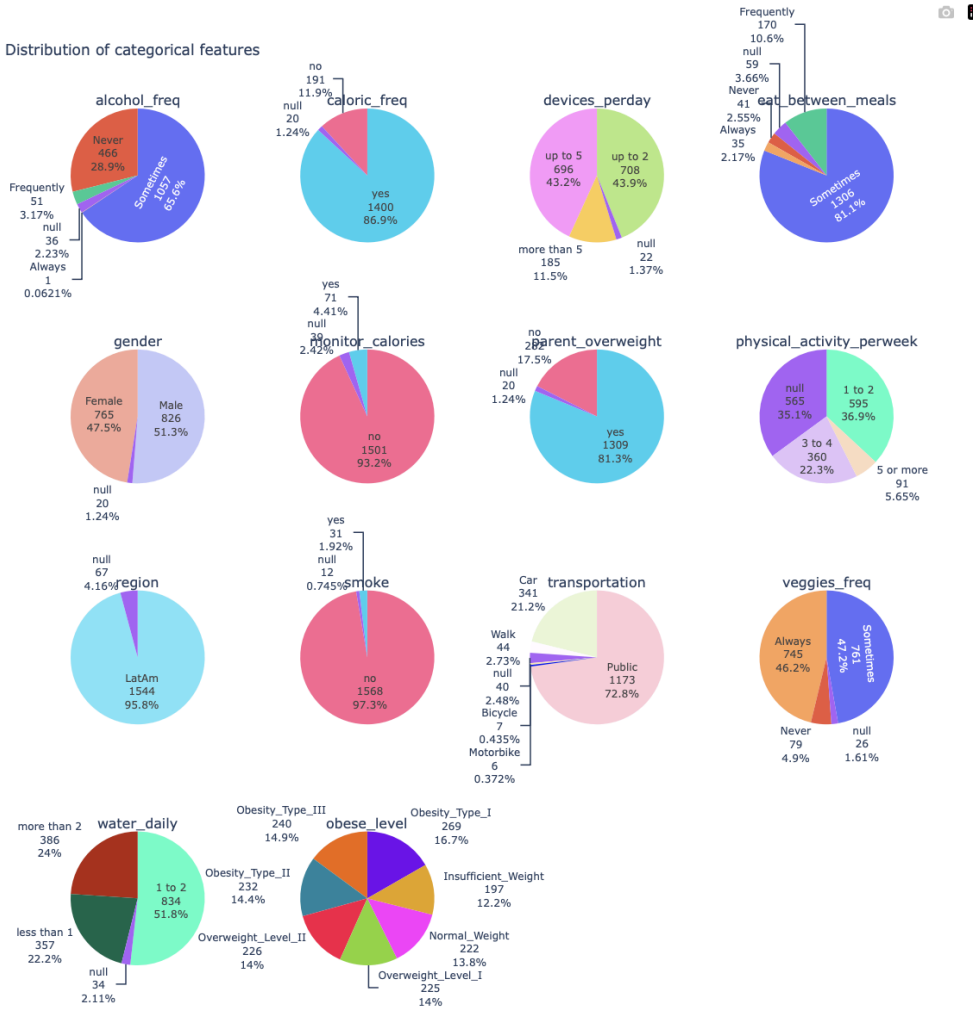
- **Age:** Ranges from 6 to 88 years with a mean of 24.35 years
- **Height:** Mean of 1.70 meters, with most values clustering between 1.6 and 1.8 meters
- **Weight:** Ranges from 32 to 193 kilograms, with a mean of 86.96 kilograms

Categorical features : *Gender, alcohol_freq, transportation, and veggies_freq* and others

- **Alcohol Frequency:** Most individuals (67.1%) report consuming alcohol "Sometimes," while only 2.2% consume it "Always."
- **Caloric Frequency:** The majority of respondents (86.9%) reported "Yes" for monitoring caloric intake.
- **Transportation:** Public transportation is the mode representing 72.8% of the responses
- **Obese Level:** The target variable contains 7 unique categories, with "Obesity_Type_I" being the most frequent class (16.7%).

The dataset also contains missing values, such as completely missing in the *marrital_status* column and partial missing values in *physical_activity_perweek* and *meals_perday mainly.*

The numerical variable distributions in the dataset reveal key insights into the participants' demographic and lifestyle patterns. Most participants are in their 20s and 30s, with heights clustering around 1.7 meters and weights between 60 and 100 kg, reflecting a younger and generally healthy population. Meal frequency predominantly centers at three meals per day, indicating a standard dietary habit, while the number of siblings suggests a balanced distribution with common family sizes.



Some observations that we can extract from the plots obtained:

- **Alcohol Frequency:** Majority (65.6%) reported consuming alcohol "Sometimes."
- **Caloric Monitoring:** Low levels observed, with 93.2% not monitoring their intake.
- **Device Usage:** Almost evenly split between "up to 5" (43.2%) and "up to 2" (43.9%) devices used per day.
- **Physical Activity:** Most participants (36.9%) engage in 1 to 2 activities per week.
- **Smoking:** Very low, with 97.3% being non-smokers.
- **Transportation Preferences:** Public transport is the most common choice (72.8%).
- **Water Consumption:** Over half (51.8%) drink 1 to 2 liters daily.
- **Obesity Levels:** Balanced distributions

This exploratory analysis highlights key patterns in the dataset, providing us some important information for predictive modelling. However, we still have to focus on some data cleaning and preparation like handling missing values and deal with feature engineering that we will address further in this report.

Methodology

1. Model assessment strategy

Our model assessment strategy was the Hold-out method. In general, cross-validation is better. In our case, we had the goal of increasing the score in Kaggle. It was our main indicator of the model. So, having a more accurate validation score was not so important. Cross-validation is more precise because the model was tested several times and the result is average. However, the Hold-out method has its limitations; which is the risk of missing important information. To address this, after identifying the best model parameters, we trained the final model on the entire dataset to optimize its performance.

2. Data Preprocessing

We started the preprocessing stage by dropping 3 columns: “martial_status” as it was completely null, “region” as we discovered that there was only one single value “LatAm”, and “siblings” as we thought that it wasn’t a good predictor of obesity.

Next, we deleted all duplicates and splitted the dataset into X_train, X_val, y_train and y_val.

Encoding

After that, we started the encoding part of the categorical variables, where we label encoded columns: 'caloric_freq', 'gender', 'monitor_calories', 'parent_overweight', 'smoke', 'transportation' as they have no specific order. For columns with a specific order, we applied ordinal encoding: 'alcohol_freq', 'devices_perday', 'eat_between_meals', 'physical_activity_perweek', 'veggies_freq', 'water_daily' with the following hierarchy:

1. *'alcohol_freq'*: ['Never', 'Sometimes','Frequently', 'Always']
2. *'devices_perday'*: ['up to 2', 'up to 5', 'more than 5']
3. *'eat_between_meals'*: ['Never', 'Sometimes', 'Frequently', 'Always']
4. *'physical_activity_perweek*': ['1 to 2', '3 to 4', '5 or more']
5. *'veggies_freq'*: ['Never', 'Sometimes', 'Always']
6. *'water_daily'*: ['less than 1', '1 to 2', 'more than 2']

To perform encoding, we used Sklearn LabelEncoder and OrdinalEncoder modules. All instances of the encoder classes, fitted on the training data, were stored in a dictionary for later use in transforming the validation and test datasets.

Important Note: Encoding was applied only to non-missing values, ensuring that missing values were not assigned any labels.

Imputation

With our categorical features encoded, we proceeded to imputation. We applied Sklearn KNN Imputer with 3 neighbors, a non-Euclidean metric, and uniform weights. The imputed values were later rounded to align with the labels / ordinal structure of the categorical variables..

Feature engineering

We decided to add a new feature called “BMI”, calculated as weight divided by height squared. This feature represents Body Mass Index, while being a “rule of thumb” is widely considered as a reliable estimator of individual obesity levels. To ensure the data relevance, we checked the scatter plot between BMI and Obesity Level and manually deleted any detected outliers.

Scaling

As a final step in our preprocessing, we scaled our data using the Sklearn module “StandardScaler”. The class instance was fitted to the train data and subsequently used to transform training, validation and later test datasets

3. Feature selection

1) We created a heat map (Screenshot in Annex). Weight and BMI have a very strong correlation. Because BMI is more correlated with the target variable, we dropped weight.
2) To evaluate feature importance, we initially used general methods like RFE and linear regression coefficients, but the results were inconsistent. Prioritizing Kaggle score, we tested feature influence directly on Kaggle. After achieving a score of 0.9565, we measured each feature's impact by shuffling it in the test data to observe the resulting drop in score.

After shuffling individual features, we observed the changes in the Kaggle score (screenshots provided in Annex). When the *BMI* feature was shuffled, the score dropped significantly to 0.1679, indicating its crucial importance to the model. For most features, shuffling had little to no effect on the score, with the exception of *Age*, which caused a slight reduction in performance.

There are 2 features—”*meals per day”* and “*eat between meals”*—that led to an increase in the score when shuffled. It shows that these features interfere with the model's ability to accurately predict the classes.

Based on this observation, we decided to remove these 2 features from the dataset.

3) We also conducted a statistical test to evaluate the correlation with the target variable. Using a chi-squared test, we analyzed the distribution and observed the p-values for each feature. None of the features fell in the normal rejection region, indicating weak statistical evidence of correlation. However, the feature with the largest p-value was “*smoke*”, suggesting it had the least relevance. Based on this result, we decided to remove the *“smoke”* feature from the dataset.

At the end we deleted 4 features from the dataset:
['weight','eat_between_meals', 'meals_perday', 'smoke']

4. Algorithms

Logistic Regression

We tested Logistic Regression for multiclass classification using two approaches:

One-vs-Rest (OvR): Trains a separate binary classifier for each class, predicting whether an instance belongs to that class or not. During prediction, the class with the highest probability is selected. **Multinomial (Softmax Regression):** Uses a single model with the softmax function to compute

probabilities for all classes simultaneously, selecting the class with the highest probability. This is more efficient and preferred for balanced datasets with inter-class dependencies. [1]

KNN Classifier

“KNN algorithms have been widely used in research and are considered to be one of the top-10 data mining algorithms”[2]. Main parameters for KNN classifiers are n_neighbors and metric. From the distance metrics Manhattan, same as Minkowski with p = 1 and Euclidean, the best performance was shown by Manhattan distance. Different scaling methods were used to analyse the performance of KNN algorithms. Based on other studies, results have shown that there isn’t a scaling method that constantly performs well with different algorithms and the best approach is to experiment with different scaling methods [6]. In this study, 3 different scaling methods were used: MinMax, RobustScaler and StandardScaler. The scaling methods can be ranked in the following order SS>RB>MM by f1-score. For this algorithm the most suitable scaling method is the StandardScaler. For the number of neighbors between 1 and 21, the best results were achieved with 3 neighbors. One of the problems for KNN algorithms is dimensionality. KNN algorithms perform worse when there are many variables. In order to fix this problem, a filter method is used that selects the features based on their statistical relationship with the target variable. SelectKBest, filter method, with mutual_info_classif, scoring function, and number of features equal to 4 finds age, gender, height, and BMI to be the most relevant features in the dataset.

Bagging Classifier with base estimator KNN

Bagging Classifier will combine multiple weak learning algorithms to reduce variance and improve overall performance. In our experimentation we implemented bagging with KNN and Decision Tree classifiers.

Bagging Classifier with base estimator Decision Tree

Two models are tested using decision trees, one with maximum depth equal to 3 (bagg_DT) and another without max_depth (bagg_DT2), giving different results. Parameters used which optimize the algorithms are n_estimators, max_features, and max_samples set to 200, 0.8 and 0.5 respectively.

Decision Trees

Gini Impurity and Entropy are both criterias used to measure the impurity in decision trees. Gini focuses on minimizing the probability of misclassification, while Entropy measures information gain by reducing uncertainty. However, the parameter that had the most significant impact on the model’s performance was max_depth, with a value of 5. Adjusting the criterion from Gini to Entropy did not lead to any noticeable changes in the results.

SVM (Support Vector Machine)

A study highlights the popularity of SVM classifiers in medical diagnosis, noting their "excellent generalization performance" [4], which supports their use in our project. SVMs minimize risk by "maximizing the margin between the separating hyperplane and the closest data point to the hyperplane" [4], ensuring low error on unseen data.

"As we expected, the effectiveness of C-SVC depends on the selection of kernel and parameters." To optimize the model, Grid Search identified the linear kernel with C=10 as the best choice, "which

allows for a certain degree of misclassification tolerance" [5]. Training with the linear kernel results in straight hyperplanes and margins, but its limited expressivity means it may not fully capture the training data's complexity.

MLP (Multi Layer Perceptron)

The MLP algorithm performs better with many hidden layers. Moreover, when applying other activation functions different from the default one which is 'relu', the algorithm performs worse. The max_iteration parameter also has an effect on the performance of the algorithm, as too few iterations will not allow the algorithm to converge, therefore for this model the optimal number was 600 iterations.

Naive Bayes

Naive Bayes classifiers are simple models that assume feature independence and apply Bayes' Theorem.

Random Forest

First, we manually tested different parameters to optimize the model, finding that *n_estimators* generally improved the robustness of predictions and including all features during training avoids removing significant predictors. Then, we used Grid Search to identify the best combination of hyperparameters for the model to determine the optimal configuration. Among the parameters to choose were: *n_estimators*, *max_depth*, *min_samples_split*, and *min_samples_leaf* as these help minimize and prevent overfitting.

Gradient boosting

Gradient Boosting is an ensemble method that builds models sequentially, where each tree corrects the errors of the previous ones, reducing bias. Key parameters include n_estimators, learning_rate and max_depth. We adjusted these parameters manually to observe their impact and used Grid Search for optimization. Random Forest was found to be the best initial model for boosting.

Stacking

Stacking is an ensemble learning technique that combines predictions from multiple models (base learners) using a meta-model. Base models are trained on the training data, and their predictions are used as input features for the meta-model, which makes the final prediction. StackingClassifier is good for our classification problem. We tried several combinations (structure is base models [] + metamodel):

1) [RF, GB, SVC, kNN] + LogReg (2 times with different models)
2) [RF, GB, SVC, bagDT, kNN] + RidgeClassifierCV
3) [RF, GB, SVC, kNN] + RidgeClassifierCV
4) [RF, GB, SVC, kNN] + GradientBoostingClassifier
5) [GB1, GB2, GB3] + [GB1, GB2, GB3] + GB
6) [RF, GB, SVC] + [RF, GB, SVC] + LogReg

The last stackings have 2 layers. All of these models have strong scores between 0.9450624197508578 and 0.9662551941777393. We noted that the best-performing combinations included different types of models as base models and a linear metamodel. Thus, we decided to add

the 4 prepared base models: [RF, GB, SVC, kNN]. We observed that stacking with only boosting models showed worse results due to the lack of diversity in the models. Also, we noticed that any non-linear model as a metamodel performs worse than linear ones because linear models balance the outputs from diverse base learners.

5. Model optimisation

Model optimization will improve models performance by fine-tuning the hyperparameters and by improving generalization.

As we already mentioned above, we used a combination of manual tuning and Grid Search to identify the best parameters. Manual tuning provided insights into hyperparameters, such as increasing the number of n_estimators for Random Forest to improve robustness and adjusting max_depth of decision trees to prevent overfitting.

Grid Search allowed us to explore combinations of parameters, such as n_estimators, max_depth, min_samples_split, and min_samples_leaf, identifying the configuration that minimized overfitting and maximized predictive accuracy. While this process was computationally intensive, it provided valuable insights into parameter influence.

We also explored stacking to combine diverse models and improve performance. Using five base models—Random Forest, Gradient Boosting, Support Vector Classifier, Decision Tree, and kNN—paired with a linear meta-model got the best F1-scores. Stacking showed the importance of model diversity, as combinations with only boosting methods underperformed due to reduced variability.

In the annexes, we include the code for Random Forest, Decision Trees, and Support Vector Classifier models, using a range of parameters through an extensive Grid Search process. While this method was time-consuming, the resulting models did not achieve the best performance.

RESULTS

We selected the F1-score macro as a main measure for evaluation since Kaggle uses it.

- ***Logistic Regression : "Comparison of the sparsity (percentage of zero coefficients) of solutions when L1, L2 and Elastic-Net penalty are used for different values of C"***

When comparing the models with different parameters of C, 0.01 and 1, and "L1" penalty, the results showed that higher value for C, reduces the penalty for the complexity of the model and allows greater flexibility. It yields a high F1-score of 0.861939.

Therefore, "we can see that large values of C give more freedom to the model. Conversely, smaller values of C constrain the model more. In the L1 penalty case, this leads to sparser solutions," meaning that more coefficients are set to 0 and only a subset of features are used to train the model resulting in lower F1-score of 0.379198.

Differences F1-score can also be seen when different regularisation techniques (penalties) are used. The highest F1-score of 0.882325 was achieved with the "L2" penalty and a C value of 10, indicating the effectiveness in balancing the model flexibility and regularisation. The model with the "L1" penalty followed closely with an F1-score of 0.860557919594379, benefitting from sparsity. The "Elastic-Net" penalty, which combines L1 and L2, resulted in the lowest F1-score of 0.85714. It may show that combined constraints can have limited the model's ability to capture more complex patterns.

- ***KNN Classifier***

For this algorithm, the most suitable scaling method identified is the StandardScaler achieving an F1-score of 0.77324. When using the reduced feature dataset, the model's performance improved significantly, with an F1-score of 0.8929, showing the importance of feature selection to boost the predictive accuracy.

Results have shown that the best validation score is achieved with 3 neighbors.

```
Accuracy score KNN Train after feature selection (k = 11):  0.9245439469320066
Accuracy score KNN Validation after feature selection (k = 11):  0.8918032786885246
F1-score KNN after feature selection (k = 11) 0.8929666230766641

Accuracy score KNN Train after feature selection (k = 3):  0.9245439469320066
Accuracy score KNN Validation after feature selection (k = 3):  0.8918032786885246
F1-score KNN after feature selection (k = 3) 0.8929666230766641

Cnclusion: KNN model performs better with less k-neighbours
```

- ***Bagging Classifier with base estimator KNN***

For the Bagging KNN model, we observed an accuracy score of 0.9618 on the training set and 0.888 in the validation set, with an F1-score of 0.890. While KNN demonstrated moderate performance, the variation between the training and validation results indicates room for improvement.

- ***Bagging Classifier with base estimator Decision Tree***

When using Decision Tree classifiers in an Ensemble Classifier the validation score improves.

Bagg_DT2 reached a higher F1-score of 0.94186 compared to Bagg_DT which had an F1-score of 0.9339.

Although the final result for both models obtained similar f1-scores, a main difference can be spotted between these 2 models. Decision trees have the tendency to overfit when there is no stopping criteria, such as max_depth. Without it, the Bagg_DT2 model's score on the training dataset was 0.98839, resulting in a larger gap between validation and training score compared to the other model (bagg_DT) with 0.9560 score for the training dataset, showing a smaller gap between the 2 scores.

This is an important situation to consider when choosing between models.

- ***Decision trees***

The F1-score of a single Decision Tree is good for predicting the obesity level. The F1-score of the decision tree is 0.9476325294719329.

- ***SVM***

In our analysis, the linear kernel achieved an F1-score of 0.932, indicating a normal performance on the validation set. The One-Vs-One (OvO) classifier, built using LinearSVC, was implemented for multiclass classification. However, this approach got a lower F1-score of 0.92683, showing that the linear kernel SVM outperformed the OvO approach in our context.

- ***Multi-Layer-Perceptron***

The MLP model got an F1-score of 0.8915662767453391 after increasing the number of iterations to 600 and using a hidden layer size of 110.

- ***Gaussian Naive Bais***

  This achieved an F1-score of 0.802080188149179

- ***Random Forest***

After increasing to more than 200 estimators, the results did not change. So, it is optimal.

Through Grid Search, the best parameters identified were {'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 200} getting an F1-score of 0.9510942828411785.

Testing additional combinations of parameters like setting criterion to 'entropy' and changing "min_samples_split" improved the F1-score to 0.9640023819296041

- ***Gradient boosting***

Default

→ F1-score 0.9416285892008212

Our parameters :

n_estimators=200,learning_rate=0.1, subsample=0.9, init= RandomForestClassifier(random_state=42, n_estimators= 200, bootstrap = True, criterion='entropy'))

→ F1-score with our parameters: 0.959957813397991

- ***Stacking***

Model	**F1-score on validation set**	**Kaggle score**
Whole grid search, code in annexes	0.9596992894967625	0,9843
[RF, GB, SVC, kNN] + LogReg with best models from the previous step	0.9641816040225583	0,9843
[RF, GB, SVC, kNN] + RidgeClassifierCV with best models from the previous step	0.9662551941777393	0,9788
[RF, GB, SVC, bagDT, kNN] + RidgeClassifierCV with best models from the previous step	0.9599586109740097	0,9843
[RF, GB, SVC, bagDT, kNN] + Gradient Boosting Classifier with best models from the previous step	0.9450624197508578	0,9665
[RF, GB, SVC, kNN] + [RF, GB, SVC, kNN] + LogReg with best models from the previous step	0.9613996800356982	0,9716
[RF, GB, SVC, kNN] + LogReg with the more simple base models	0.9598805098120915	0,9843
[GB1, GB2, GB3] + [GB1, GB2, GB3] + LogReg	0.9490605855865358	0,9565
final, trained with all data [RF, GB, SVC, kNN] + RidgeClassifierCV with best models from the previous step	no validation	0,9843

We can see that all models show great results, with some achieving slightly higher results than others. However, in the test dataset, we still have some instances where individuals do not align with the rules of our model.

Discussion

This study highlights the effectiveness of machine learning in predicting obesity levels, leveraging personal and lifestyle characteristics to deliver meaningful insights. Among the tested models, stacking emerged as the most effective, achieving a high Kaggle score. By combining diverse base models, including Random Forest, Gradient Boosting, and Support Vector Machines, stacking demonstrated its ability to balance model strengths and minimize weaknesses.

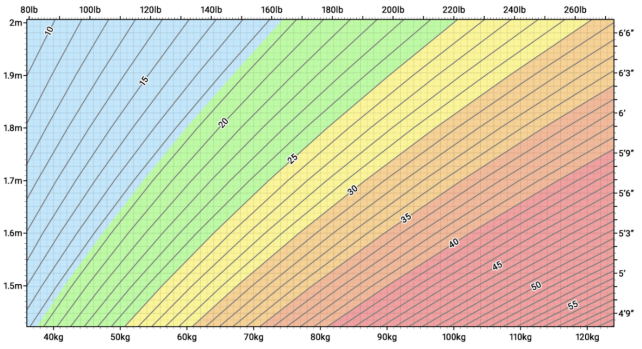
Key features such as BMI, age and physical activity levels were identified as significant predictors of obesity. The importance of BMI aligns with existing research emphasizing its reliability as a metric for assessing obesity levels [3]. This result validates the decision to engineer BMI as a feature and remove redundant variables like weight. The influence of age and physical activity on the model's performance also resonates with the broader literature, which frequently associates these variables with obesity trends. Shuffling experiments further reinforced the importance of these features, with BMI showing the largest impact on model performance when disrupted, emphasizing its critical role in obesity classification.

Despite these promising results, the study faced limitations due to the scope of algorithms available in scikit-learn. While neural network models are available in scikit-learn, such as Multi-Layer Perceptrons (MLPs), their performance was not competitive enough to surpass the other tested algorithms. Advanced architectures which are known for capturing complex, non-linear relationships, could not be explored. Incorporating these models in future work could further enhance predictive accuracy and adaptability. The no free lunch theorem underscores the importance of testing a range of algorithms, as no single model excels universally across all datasets.

Interestingly, we also discovered that a simple formula based solely on BMI could achieve comparable performance to the machine learning models on Kaggle (0.9792). While this approach is computationally efficient, it may lack the flexibility required for broader datasets.

The excel formula applied to every observation and distribution of obese level by BMI (Wikipedia):

```
=IFS(
 R2 < 18,5; "Insufficient Weight";
 R2 < 25; "Normal Weight";
 R2 < 30; "Overweight Level I";
 R2 < 35; "Overweight Level II";
 R2 < 40; "Obesity Type I";
 R2 < 45; "Obesity Type II";
 R2 >= 45; "Obesity Type III"
)
```



These findings highlight the importance of balancing simplicity and flexibility when developing predictive models.

Conclusion

This study demonstrated the potential of machine learning in addressing public health challenges, focusing on the prediction of obesity levels. Through careful feature engineering and model evaluation, stacking emerged as the most effective approach, highlighting the value of combining diverse algorithms. Key factors such as BMI, age, and physical activity were confirmed as significant predictors, aligning with established research.

While simpler approaches, such as BMI-based formulas, showed comparable performance in specific contexts, machine learning models provide greater flexibility and adaptability for broader applications. However, the study was limited by the scope of algorithms available, suggesting that future research should explore advanced methods and more diverse datasets to enhance generalizability.

This project provides a foundation for integrating machine learning into public health, demonstrating its ability to uncover meaningful insights and support data-driven interventions. Future work should build on these findings to refine methodologies and develop scalable solutions to combat obesity globally.

References

[1] https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html

[2] Shichao Zhang, Senior Member, lEEE ; Challenges in KNN Classification https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9314060

[3] Abu Alfeilat HA, Hassanat ABA, Lasassmeh O, Tarawneh AS, Alhasanat MB, Eyal Salman HS, Prasath VBS. Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. Big Data. 2019 Dec;7(4):221-248. doi: 10.1089/big.2018.0175. Epub 2019 Aug 14. PMID: 31411491.

[4] J. Novakovic and A. Veljovic, "C-Support Vector Classification: Selection of kernel and parameters in medical diagnosis," 2011 IEEE 9th International Symposium on Intelligent Systems and Informatics, Subotica, Serbia, 2011, pp. 465-470, doi: 10.1109/SISY.2011.6034373. keywords: {Kernel;Accuracy;Polynomials;Diabetes;Cancer;Liver;Medical diagnostic imaging;C-SVC;classification accuracy;kernel;parameter selection;SVM},

[5]https://scikit-learn.org/1.5/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-p

[6] Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review

ANNEXES

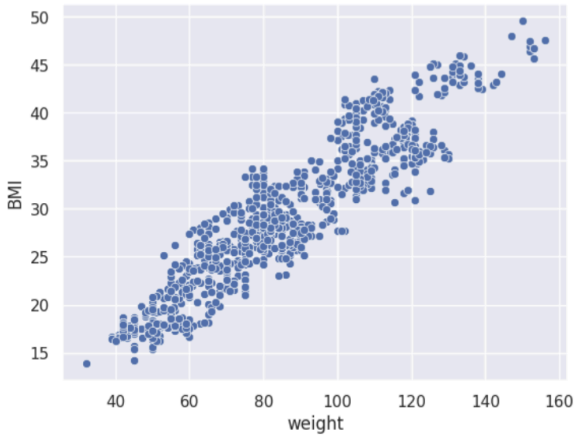
(Step 1: Data exploration)

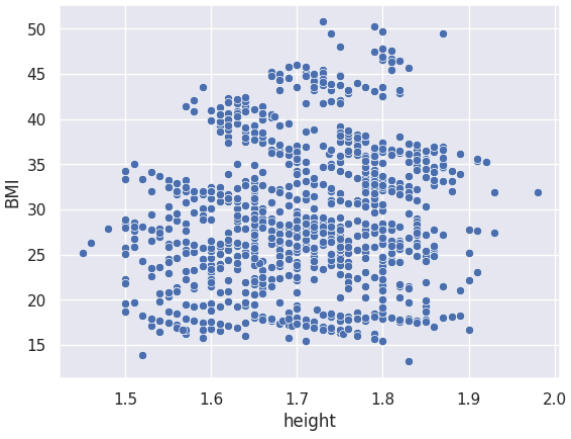
	count	mean	std	min	25%	50%	75%	max
age	1545.0	24.344984	6.474498	6.00	20.00	23.0	26.00	88.00
height	1597.0	1.704108	0.095567	1.29	1.63	1.7	1.77	2.19
marrital_status	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
meals_perday	1602.0	2.684145	0.817584	1.00	3.00	3.0	3.00	4.00
siblings	1599.0	1.500938	1.132562	0.00	0.00	2.0	3.00	3.00
weight	1558.0	86.956354	26.072339	32.00	67.00	83.0	107.00	193.00

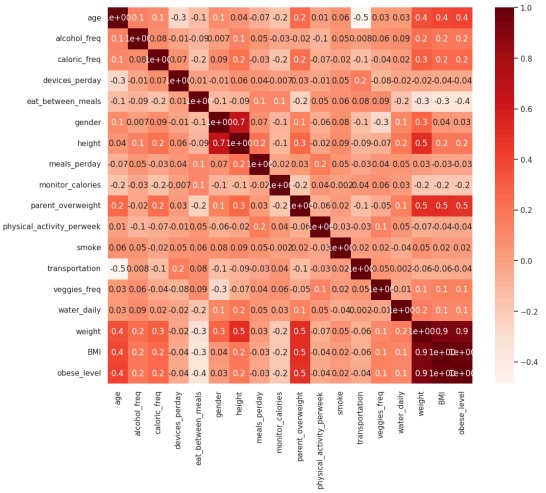
	count	unique	top	freq
alcohol_freq	1575	4	Sometimes	1057
caloric_freq	1591	2	yes	1400
devices_perday	1589	3	up to 2	708
eat_between_meals	1552	4	Sometimes	1306
gender	1591	2	Male	826
monitor_calories	1572	2	no	1501
parent_overweight	1591	2	yes	1309
physical_activity_perweek	1046	3	1 to 2	595
region	1544	1	LatAm	1544
smoke	1599	2	no	1568
transportation	1571	5	Public	1173
veggies_freq	1585	3	Sometimes	761
water_daily	1577	3	1 to 2	834
obese_level	1611	7	Obesity_Type_I	269

```
<class 'pandas.core.frame.DataFrame'>
Index: 1611 entries, 1 to 1611
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   age                         1545 non-null   float64
 1   alcohol_freq                1575 non-null   object
 2   caloric_freq                1591 non-null   object
 3   devices_perday              1589 non-null   object
 4   eat_between_meals           1552 non-null   object
 5   gender                      1591 non-null   object
 6   height                      1597 non-null   float64
 7   marrital_status             0 non-null      float64
 8   meals_perday                1602 non-null   float64
 9   monitor_calories            1572 non-null   object
 10  parent_overweight           1591 non-null   object
 11  physical_activity_perweek   1046 non-null   object
 12  region                      1544 non-null   object
 13  siblings                    1599 non-null   float64
 14  smoke                       1599 non-null   object
 15  transportation              1571 non-null   object
 16  veggies_freq                1585 non-null   object
 17  water_daily                 1577 non-null   object
 18  weight                      1558 non-null   float64
 19  obese_level                 1611 non-null   object
dtypes: float64(6), object(14)
memory usage: 264.3+ KB
```

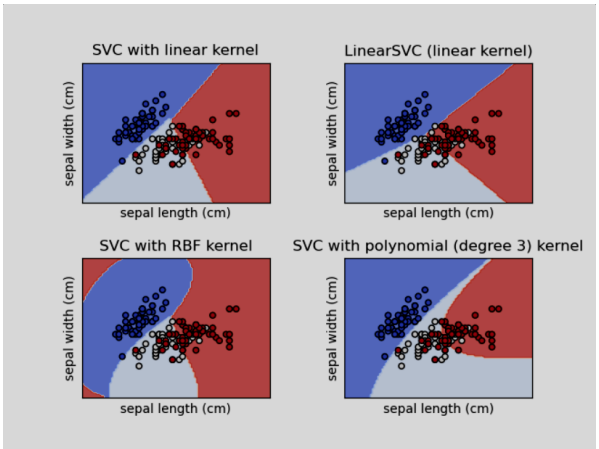
(Step 2: Correlation between variables)







(Support Vector Machine)

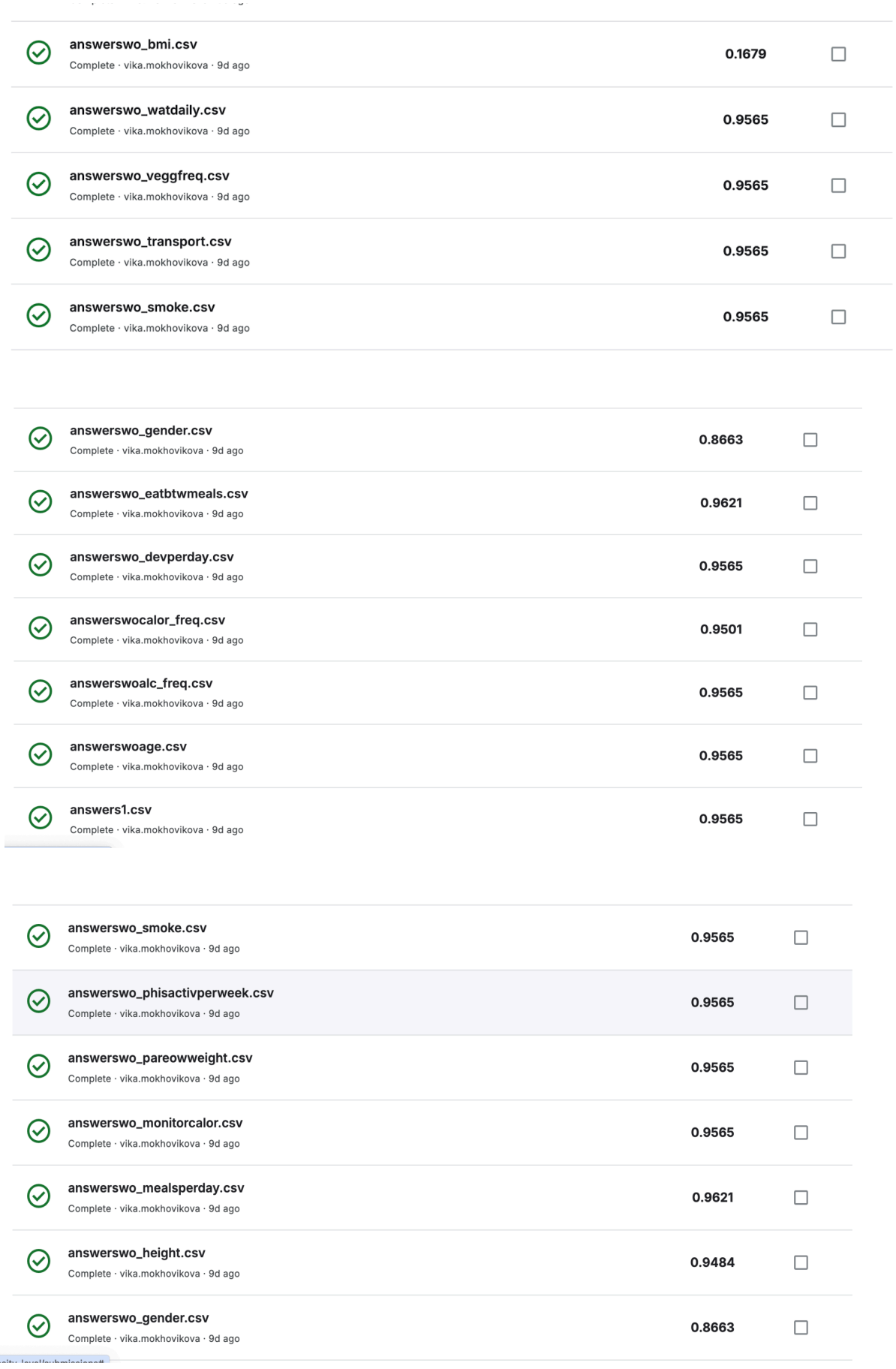


(Grid Search code)

```
gb_best = GradientBoostingClassifier(
    ccp_alpha=0.0,
    criterion='friedman_mse',
    learning_rate=0.05,
    loss='log_loss',
    max_depth=5,
    min_samples_leaf=5,
    min_samples_split=8,
    n_estimators=200,
    subsample=0.6,
    tol=0.0001,
    validation_fraction=0.1,
    verbose=0,
    warm_start=False
)
svc_best = SVC(
    C=10,
    break_ties=False,
    cache_size=200,
    coef0=0.0,
    decision_function_shape='ovr',
    degree=3,
    gamma='scale',
    kernel='linear',
    max_iter=-1, probability=True, tol=0.001, verbose=False,shrinking=True
)
rf_best = RandomForestClassifier(
    bootstrap=True,
    min_samples_leaf=2,
    min_samples_split=5,
    n_estimators=200,
    max_features='sqrt',
    random_state=42
)
```

```
param_grid_rf = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_leaf': [1, 2, 4],
    'min_samples_split': [2, 5, 10]
}
grid_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid_rf, cv=3, scoring='f1_macro')
grid_rf.fit(X_train, y_train)
rf_best = grid_rf.best_estimator_
param_grid_gb = {
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 5, 8],
    'min_samples_leaf': [1, 3, 5],
    'min_samples_split': [2, 8, 10],
    'n_estimators': [100, 200],
    'subsample': [0.6, 0.8, 1.0]
}
grid_gb = GridSearchCV(GradientBoostingClassifier(), param_grid_gb, cv=3, scoring='f1_macro')
grid_gb.fit(X_train, y_train)
gb_best = grid_gb.best_estimator_
param_grid_svc = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf']
}
grid_svc = GridSearchCV(SVC(probability=True), param_grid_svc, cv=3, scoring='f1_macro')
grid_svc.fit(X_train, y_train)
svc_best = grid_svc.best_estimator_
```

(Kaggle Submissions)



NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa