

Computer Science Capstone - C964

Kevin Salazar

Student ID: 010303855

March 6, 2024.

Contents Table

Task 2: Part A, B, C and D

Part A: Letter of Transmittal.....	3-4
Part B: Project Proposal Plan.....	5
Project Summary.....	5
Data Summary.....	5
Implementation.....	6
Timeline.....	7-9
Evaluation Plan.....	7-9
Resources and Costs.....	7-9
Hardware and software costs.....	7-9
Labor costs.....	8-9
Environment costs.....	9
Part C: Application.....	10-27
Machine Learning.....	18
Visualizations.....	24-27
Part D: Post-implementation Report.....	28
Solution Summary.....	28
Validation.....	29
Data Summary.....	30-32
User Guide.....	36-38
References.....	39

Part A: Letter of Transmittal

March 6, 2024

Kevin Salazar

Chief of Technology of StriveLife

Thomas Grande,

CEO of StriveLife

Palo Alto, California

Esteemed Thomas,

My reason behind this message is the following: as a chief of technology it is my duty to innovate the structure of our company's technology sector.

StriveLife has been in business since the early 2010s, initially focusing on the sale of health plans (meal plans, fitness plans), as the company grew, the linkage of healthcare providers for specific plans was offered. Later on, around 2016, as digitalization became more prominent, the tracking abilities within the company app was an option for all consumers, and this was a huge boost for StriveLife, people were finally able to track how much they eat, sleep and walk.

As the years went on, with social media becoming the primary source of information and decision making for the average consumer, people are recognizing the importance of obesity level for health purposes, social media is bringing awareness of how being at a healthy weight can improve and even expand your lifespan. Multiple competitors to the company have risen, AI has become a controversial topic generating a lot of attention from the public, making major headlines in mass media communications. With this information, it is a must for the company to develop a tool that involves some kind of AI, to benefit itself in the current technology and market conditions. A remedy for this is, with the help of machine learning methodologies, to develop a Physical Fitness Level predictor, that would help users know where they stand at with their physical health, and predict any other results based on the modification of fitness metrics.

The idea is a web application that takes input from the user, after the input is taken, a trained model will perform the prediction of their fitness level based on their current fitness metrics (Age, Gender, Weight, Height, BMI and Physical Activity Level). The prediction will let the user know what Fitness Level they are at. This will also allow the user to modify their weight to a target weight, to see what is their ideal weight according to their personal metrics, predicting the results they could have once reaching such results of the prediction in real life.

Following the completion of this prediction tool, user engagement with our health platform will increase drastically, as it involves some kind of Artificial Intelligence users will promote it on social media, customer satisfaction will generate great reviews, consumer loyalty will increase, more health plans will be purchased and insurance claims thorough our networks will be made, displaying a better brand image and therefore more improvement and growth for our health and fitness plans.

If the tool is successful, a company revenue growth of 15% to 25% can be estimated, cancellation rates will drop 13%, and new memberships should increase 10%, majority of the revenue will come from helping the customer make a decision of purchasing one of the health plans from the company, or contact a healthcare provider from our network for any medical necessities that they could need in order to reach a specific goal.

A prototype will be submitted for your review after this message is acknowledged, thank you for considering this change for the good of the company.

Best wishes,
Kevin Salazar

Part B: Project Proposal Plan.

The **decision support problem and opportunity** that the project aims to solve the difficulty of trying to figure out what fitness level they are at, which can be difficult if the right tools are not gathered, it eases the process for the individual by just entering a single input of their current fitness metrics (relevant information), with an accurate result of where they stand at (or where could they be at if they reached a specific goal). When it comes to the opportunity, we can say that after the usage of this tool, we are able to capitalize on the next steps from the user, once they are aware of where they stand at, they will more likely choose either one of our health plans or healthcare providers from our networks, linked to the data from results and desired goals they have.

Our **customers consist of individuals** who want to better their quality of life, by working on their food consumption, fitness level and weight-loss, these individuals may have limited time in their life to go and have a check-up at a health facility, so a digital tool that they are able to use from the comfort of their house will save time and effort, and then help them choose a specific plan, it cannot all be done at their desktop. After following the health plans chosen, they should **accomplish desired health goals**.

The **data product** will prioritize the usage of already existing data of customers in the company, only the fitness metrics of those who have been eagerly tracking progress throughout the years with our company. We estimate an amount of 150 million users in our database, so in order to make a demo of the prototype and save time on processing and building the model before approval, an open source dataset with less volume of data will be used. The mentioned open source dataset, can be found in the following link:

<https://www.kaggle.com/datasets/mrsimple07/obesity-prediction/data>

Methodology and Implementation

The **desired methodology for this project is the Agile methodology**. The reason for choosing this methodology is the focus on iteration, this is able to help changing requirements, this project is created based on customer feedback and collaboration (initial surveys and the data gathering).

The **deliverables associated** with the data product are the prediction output of the Fitness/Obesity Level category of the user. This way, the customer can be aware of their fitness level, and this information is of aid when choosing one of the services that the company provides.

The project **development** will initiate with **planning and gathering the requirements**. This is not data gathering for the prototype, as it will come from an open source. All the data gathering for the data product will be performed in this phase, the cost of gathering data from different individuals in our database from a wide diversity of backgrounds, and this should cost up to \$2,000 USD, cost may vary depending on the complexity found by developers within the company, avoiding any data that can have gaps, for better accuracy and results.

In the next step of the planning process, the **data will be pre-processed** for future implementation. This involves developers **encoding** any necessary labels for easier reading in the machine learning model, **splitting** the data into a training set and a test set, and then **training** those sets of data; an estimated cost for this would take in mind the large data volume of over 150 million users located in the company's database, ranges between \$30,000 to \$40,000 USD.

After this, the **design of the web app** ranges from 2 to 6 weeks, the cost will vary according to simplicity and usability for the user experience, expecting a cost of \$6,000, an early model prototype version of the web app will be attached for your review, focusing on functionality.

Two machine learning models will be used in the prototype tool for testing purposes in **finding the best accurate model**, for a better **outcome**. If one of the models is not predicting correctly enough, it will be discarded from the web prototype.

An estimated 15% of the total **costs** will go into testing, it should take from 1 to 2 weeks for testing to be completed.. Principal costs regarding the project will be based on labor, an approximate number 180 hours of software engineers is needed, costing roughly \$30,000 USD

In order to ensure that the **data product meets the requirements** stated for the customer, the following will be done:

At the **early stages of this data product idea**, a set of surveys were deployed in the company website, gathering information from present users, asking about possible improvements.. These results are the backbone of the data product, due to its focus on customer needs. Customers had a positive score on a possible tool that can let them know more about their current health, and make predictions for future changes; out of 5000 surveys made, an estimate of 9 of 10 users were interested in the possible proposal. The surveyed users were a mix of all users in the company; loyal status, frequent and not so frequent users were surveyed. Proving verification of the requirements needed in order to **satisfy user needs**.

The **programming environment for this prototype** will be using **Jupyter Notebooks**, an open source tool for data processing, involving live coding and representation of media. **Google Colab**, will host the notebooks; this the environment where the prototype will be built. Free of cost as they are both open source tools. The initial **human resources** come from the survey testing done, a portion of the dataset contains the information of those users that took the survey. **Machine learning libraries** and tools such as 'pandas', 'numpy', 'streamlit', 'pickle' and others will be imported into the Notebook as needed for the data cleansing, splitting, training, representation and accuracy representation of the models. Once the **prototype is approved by executives**,

an estimate of \$5,000 will go into programming environments and related costs, using the preferred environment for current StriveLife developers.

Timeline for the final product: including milestones, start and end dates, duration and costs needed for assigned tasks.

Milestone	Duration	Start/End date	Task	Cost of resources
Planning and requirement gathering	1.5 Sprint = 3 weeks	April 1st to April 21st, 2024	Gathering dataset, programming environment	\$5,000
Design	1 Sprint = 2 weeks	April 21st to May 6th, 2024	Designing the web app	\$6,000
Development	1.5 Sprint = 3 weeks	May 6th to May 27th, 2024	Preprocessing data, machine learning model creations, web app development	\$33,000 USD (Labor and Programming Environment)
Testing	1 Sprint = 2 weeks	May 27th to June 10th, 2024	Testing code from ML models, making it accurate	\$5,000
Deployment and Launch	0.75 Sprint = 1.5 weeks	June 10th to June 19th, 2024	Launching the data product into the company user site.	\$10,000
Review	0.75 Sprint = 1.5 weeks	June 19th to June 28th, 2024	Monitoring launch	\$5,000
Total	13 weeks			\$64,000

Part C: Application

The files required for the execution of the application are the following:

- A text file called '**ReadMeFirst.txt**', with instructions on how to run the application.

This will thoroughly explain how to run the web application to achieve the proposed results.

- The jupyter notebook, named '**obesidadPrediction.ipynb**', which can be imported to a google drive and opened in Google Colab. It contains the import of necessary libraries to create the ML model for the data product; it also contains plotting of graphics, describing statistics related to the dataset, this is the **descriptive method**. Near the end of the notebook, 'Streamlite' is used to create and host the website for the prototype of the data product.

- The main application Python file named '**paginaDePredict.py**', which must be imported into the notebook to run the prototype web app, this contains all GUI elements. And it is run within one of the cells inside the notebook.

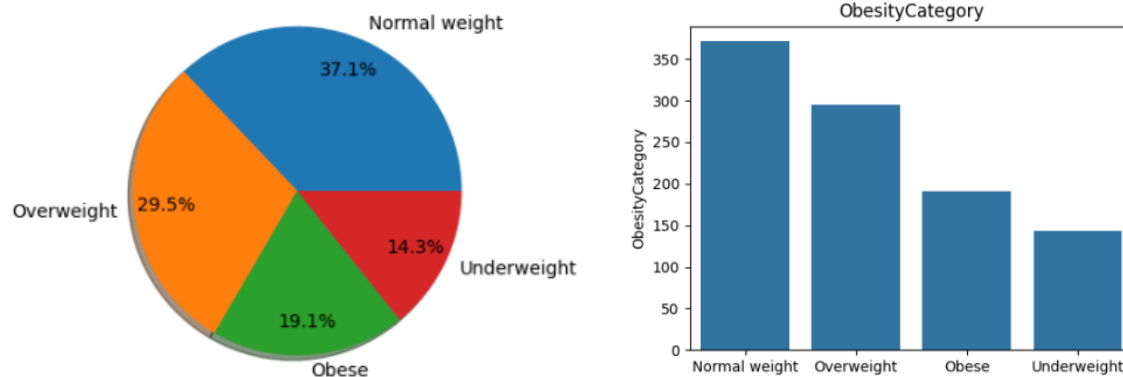
- The csv file named '**obesity_data**', this is the data frame used. It contains six features: Age, Gender, Height, Weight, BMI and Physical Level. With one label that will be used for prediction, Obesity Category.

The descriptive method(s) presented will be the following:

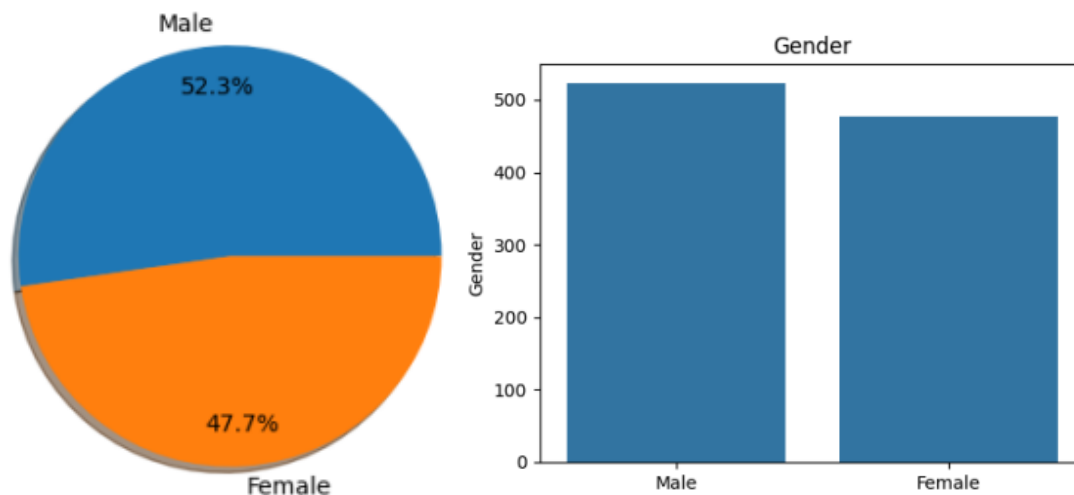
Graphical representation in the form of a **Histogram**, displaying the **relation between all 5 features** from the dataset: BMI, Weight, Height, Age, and PhysicalActivity level, showing how many individuals (Count) fall within those categories. Making the user aware of how predominant each individual feature is in the dataset, in relation to the count of information. The line represents the density of the data distribution.



The next **descriptive representation** is a Pie Chart and a Histogram displaying percentage and amount of individuals for each one of the Fitness/Obesity category levels in the dataset, the user can find this useful because it would let them know their place compared to everyone else in the dataset, all categories were used in the training of the model, for accurate results.

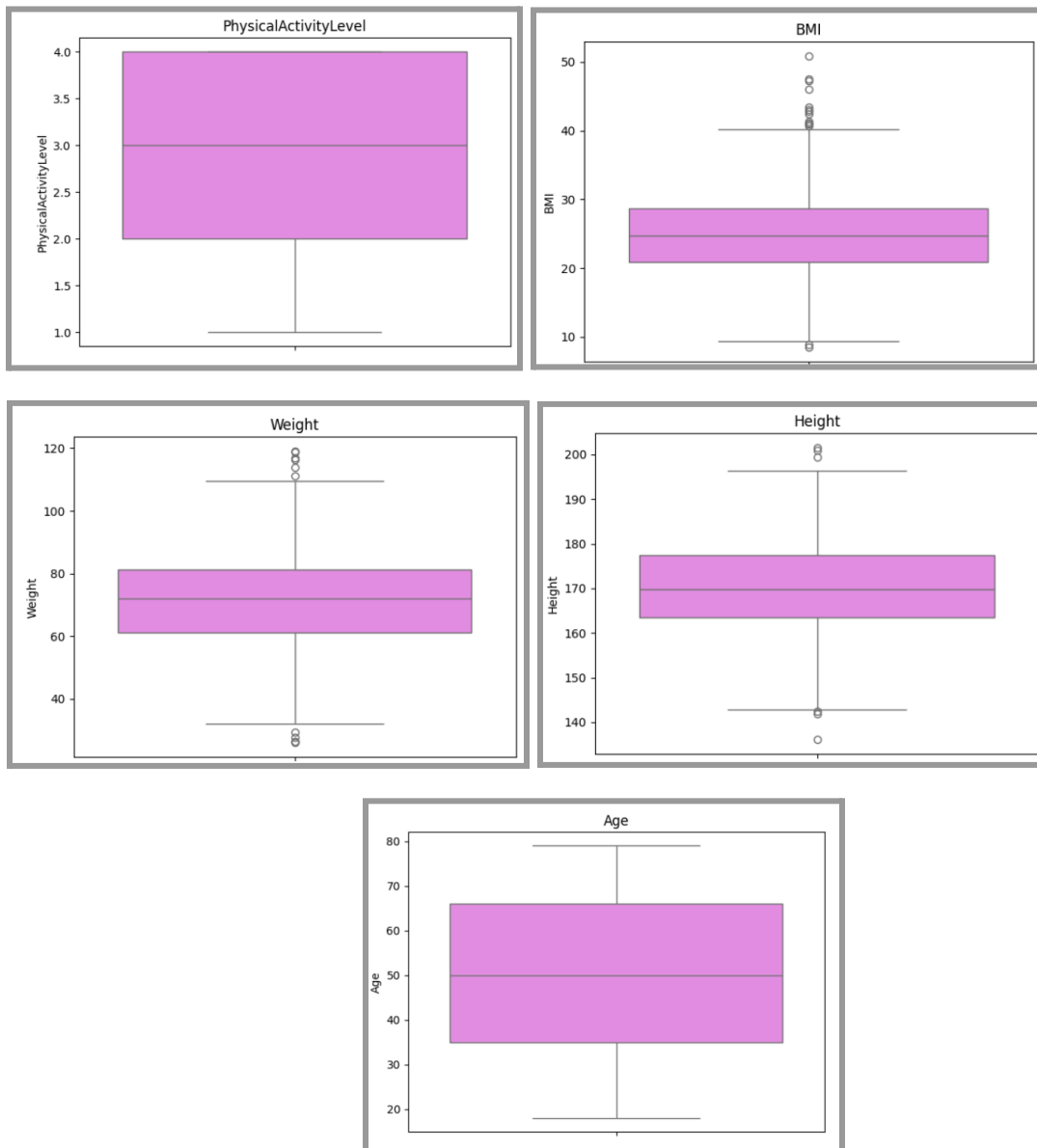


The next **graphical demonstrations** are linked to the relation between gender encountered in the dataset, there is an even distribution of both male and female individuals that took part in the dataset.

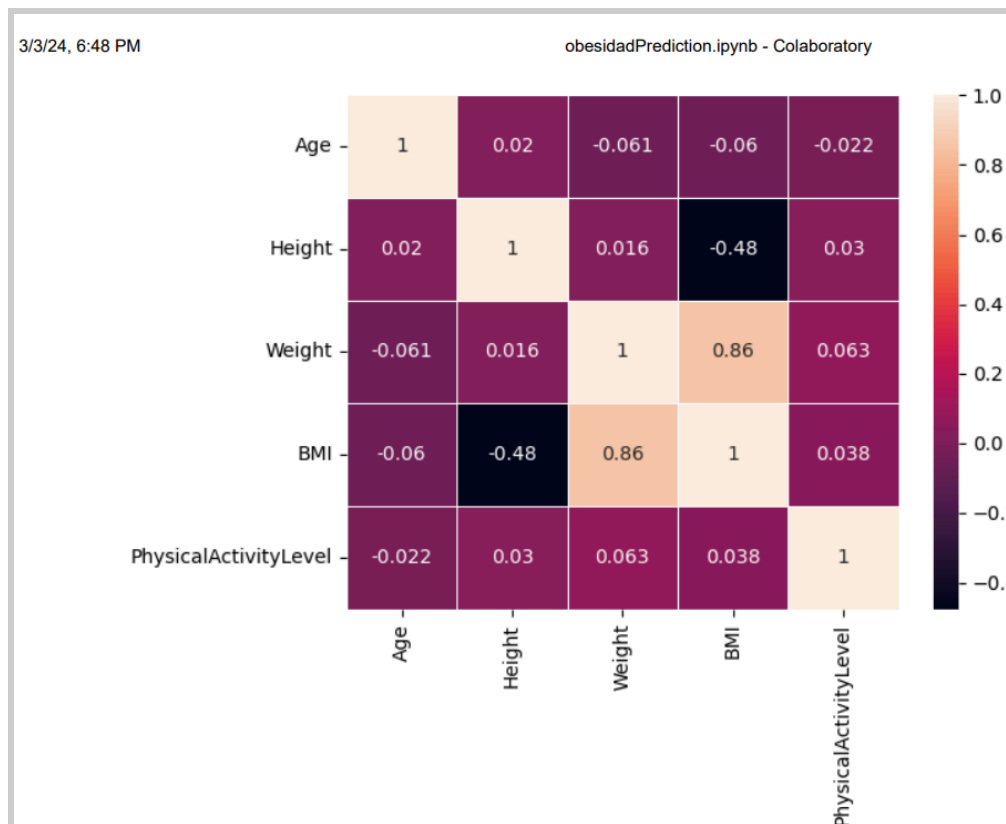


The following **graphical representations** are **box plots** linked to the existing features in the dataset, PhysicalActivityLevel, BMI, Weight, Height and Age. A box plot is read

the following way: it includes a rectangular shape or “box”, this box represents the location of the middle portion of all data, this is called **interquartile range**. The first quartile is the mark that begins the perimeter of the box, the median is represented by the line that divides the box in half and the upper part of the box is the third quartile of the data. The white circles that fall outside the boxes and outside the clashing lines (that are referred as “whiskers”) represent “outlier” data; if they were included inside the box plot, they could affect the representation of the **interquartile range** and the **median line**.



The **following graph** represents a **heat map**; with the use of colored squares, it depicts values for the variables of interest (in this case all 5 features of the dataset, gender is not included) across two axes. **The correlation scale** (the bar on the right) goes from +1 to -1, and its color coded for easier visualization. The **diagonal squares** that are of a light beige color are perfectly correlated, this is because they match with the same variable (let's say, in the matrix, 'Age x Age' equal 1 due a match on the same variable itself). **All color boxes** that are near to 0, represent a non existent linear trend between the two variables. **The color boxes** near to 1 mean that they are positively correlated. **The colored boxes** in the negative ranges have an opposite or negative correlation, similar to the positive correlation but the difference is that instead of both variables increasing, one of them decreases while the other one increases. The representation turned out to be symmetrical and the reason for this is because the same 5 variables are being plotted against themselves.



The non-descriptive method in this application is the Prediction form that outputs a result, the result lets the user know which fitness category they fall in, here is an example of a prompt, inside the Notebook. The np.array() takes an input array of all features in the dataset. The user can modify their weight to predict a target weight that they may reach in the future to fall under a specific category.

```
[ ] # Making a new prediction after model was trained
X = np.array([[22,1,180.22,70.33,22.22,4]])
X_new_scaled = scaler.transform(X)

[ ] # Printing the prediction output
y_pred = model.predict(X_new_scaled)
y_pred

array(['Normal weight'], dtype=object)
```

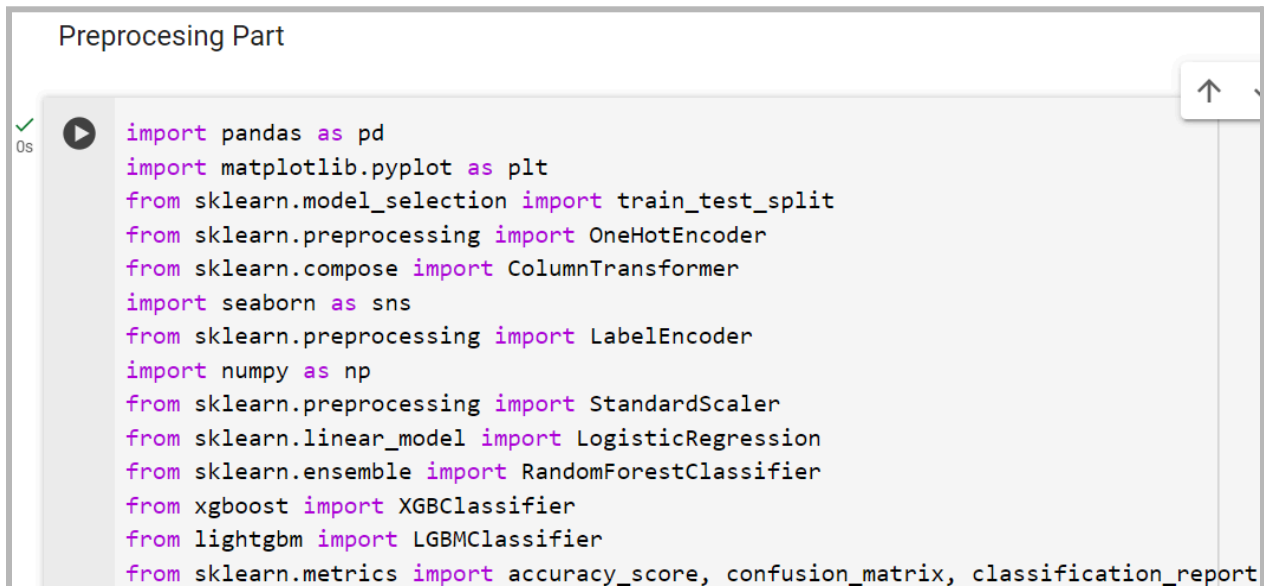
The available dataset in the prototype is the following, in the next image, the “head()” function was used to display the initial 5 rows of it.

```
setDatos = pd.read_csv('obesity_data.csv')
```

```
setDatos.head()
```

	Age	Gender	Height	Weight	BMI	PhysicalActivityLevel	ObesityCategory
0	56	Male	173.575262	71.982051	23.891783	4	Normal weight
1	69	Male	164.127306	89.959256	33.395209	2	Obese
2	46	Female	168.072202	72.930629	25.817737	4	Overweight
3	32	Male	168.459633	84.886912	29.912247	3	Overweight
4	60	Male	183.568568	60.038045	20.187003	3	Normal weight

Data features, parsing, cleaning, and wrangling datasets



```
Preprocessing Part

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

The image above represents all importation of libraries and methods needed for the execution of this prototype:

- Pandas – Open source tool for data analysis and manipulation.
- Matplotlib.pyplot – Tool that aids in the plotting of data.
- Train_test_split – Divides data into training and testing subsets.
- OneHotEncoder – Aids in the creation of dummy variables.
- LabelEncoder – Utility that transforms non-numerical values to numerical.
- ColumnTransformer – Used in creation and application of transformers for categorical and numerical data.
- Seaborn – Library for statistical graphic representation of data.
- StandardScaler – Tool used to standardize input data so data points can achieve a balanced scale.
- LogisticRegression – Supervised Machine Learning Algorithm, primary focus is to predict categorical outcomes.
- RandomForestClassifier – Supervised ML Algorithm, used to solve regression and classification. This method combines predictions from other models, making it an ensemble method.
- Accuracy_score – Helps calculate accuracy score between predicted labels and true labels.

- Confusion_matrix - ML measurement that classifies outputs that could be two or more classes.
- Classification_report - Method that displays classification metrics on a per-class basis.
- Pickle - Tool used to save trained ML models.

Defining X and y

```
le = LabelEncoder()
setDatos["Gender"] = setDatos[["Gender"]].apply(le.fit_transform)

y = setDatos["ObesityCategory"]
X = setDatos.drop("ObesityCategory", axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

The image above represents the **cleaning, training and splitting of the data**, Beginning with encoding all non-numeric values for easier processing. After encoding values, X and y were defined. Data was split between all X and y variables (both train and test). At the end there was a scaler used to determine the parameters of the dataset, fit to data and then transform it.

Machine Learning Models and Predictions

In the development of this prototype, **two models** were selected for optimal selection and greater results. In the image below, a creation of a list that includes the following machine learning models: **RandomForestClassifier** and **LogisticRegression**.

The **reason** behind the selection of the **Random Forest classifier** model is that because it is ideal for multiple quantities of features, regular linear regression is not able to represent all features in a graphical prediction due to the size of the dimensions when defining X and y; it also uses **ensemble learning**, which is a technique that uses weak learners (multiple decision trees) to make more accurate predictions, without making assumptions on the data.

Logistic Regression was selected because it is well-known and widely used, and its fast training and prediction abilities, it also has low variance so it is less likely to act sensitive in case of training data changes.

A loop passes through the lists to train the models, printing results of accuracy from each model at the end of the code block.

```
3/3/24, 6:48 PM                                obesidadPrediction.  
import warnings  
warnings.filterwarnings("ignore")  
  
model_list = [RandomForestClassifier(), LogisticRegression()]  
  
model_name_list = []  
model_accuracies = []  
  
for i in model_list:  
    model = i.fit(X_train_scaled, y_train)  
    model_name = model.__class__.__name__  
    y_pred = model.predict(X_test_scaled)  
    accuracy = accuracy_score(y_test, y_pred)  
  
    model_name_list.append(model_name)  
    model_accuracies.append(accuracy)  
  
    print(f"{model_name} accuracy: {accuracy:.3f}\n")  
  
    RandomForestClassifier accuracy: 0.996  
  
    LogisticRegression accuracy: 0.956
```

As it can be seen the **accuracy for RandomForestClassifier is close to 1, at 0.996**, and the **LogisticRegression model has a slightly lower accuracy of 0.956**.

The **models were saved** using 'pickle'. For prediction use inside the web application.

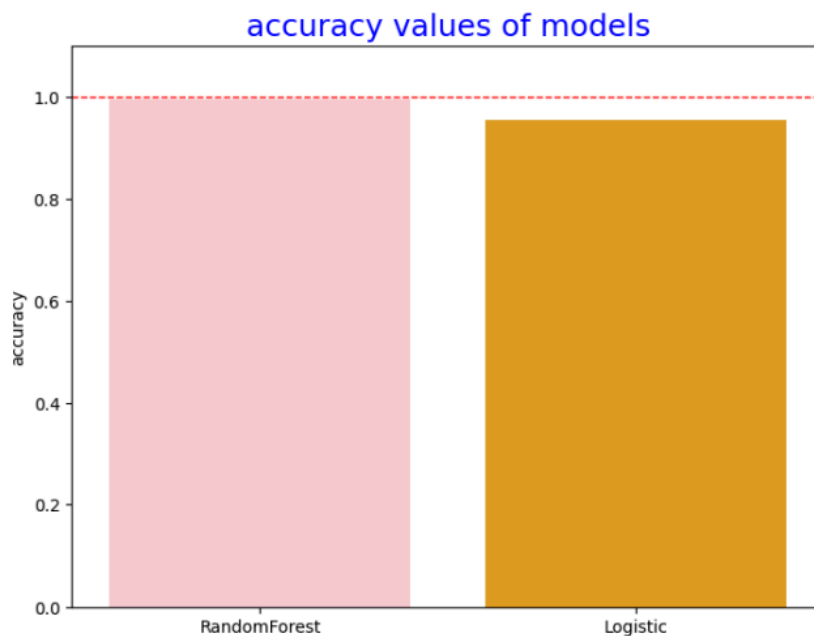
```
3/3/24, 6:48 PM obesidadPredic  
  
import pickle  
# Save trained models with pickle  
for model, model_name in zip(model_list, model_name_list):  
    with open(f"{model_name}.pkl", "wb") as f:  
        pickle.dump(model, f)
```

Validation with Graphical Representations of Model Accuracy

Histogram Representation

In the following image, a cell used to display previous accuracy results is run to display a **histogram representation of accuracy**. **Both models above 90% accuracy**. Giving promising results when implemented in the future data product.

```
fig, ax=plt.subplots(figsize=(8,6))  
cols = ["orange" if i < (max(model_accuracies)) else "pink" for i in model_accuracies]  
sns.barplot(x=model_name_list, y=model_accuracies, ax=ax, palette=cols)  
plt.ylim(0, 1.1)  
plt.ylabel("accuracy")  
plt.axhline(1, lw=1, ls="--", color="red")  
ax.set_xticklabels(["RandomForest", "Logistic"])  
plt.title("accuracy values of models", fontsize=18, color="b")  
plt.show()
```



Random Forest Classifier Confusion Matrix Representation with Test Data

The following cell executes to **create a confusion matrix** for the Random Forest algorithm, applied to the test data from the data set.

```
rf = RandomForestClassifier().fit(X_train_scaled, y_train)
y_pred_rf = rf.predict(X_test_scaled)

plt.figure(figsize=(8,5))
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, linewidths=0.6, cmap="viridis", fmt='.2g')
plt.title("RandomForest Confusion Matrix", fontsize=18)
plt.show()
```

This graph type gives a set of squares that offer different analogies, which are:

- True positive (TP): When you predict positive and it is indeed true.
- True negative (TN): Negative is predicted, and it is negative indeed.
- False positive (Type 1 Error): Predicting positive and it is not true.
- False negative (Type 2 Error): Predicting negative and it is not true.

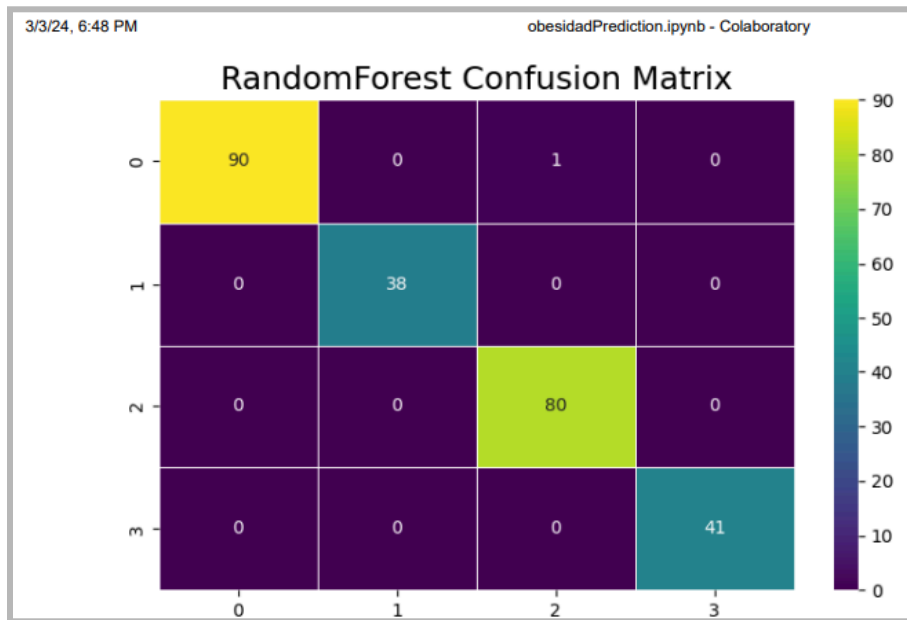
In the **diagonal**, the squares of a lighter color, located in class 0 and 2 were predicted more times than 1 and 3. The **diagonal made of blue-green-yellow colors** represents all true predictions, in all 4 fitness category levels. **Majority of the test data was predicted to be true**, except for that False positive in the upper section, where the purple boxes are, a **single error, a type 1 Error** (the box is brown and has a 1 on it). The almost perfect accuracy was achieved.

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Following these formulas we get a **recall of 1** ($TP = 128 / TP + FN = 128$), and a **precision of 0.99224** ($TP = 128 / TP + FP = 129$). The following page shows the representation of the confusion matrix.

Representation of the Confusion matrix for the Random Forest algorithm used.



Implementation of interactive queries

The following image displays the dataset inside the web app, the user is able to query and search similar metrics inside the dataset used for the prototype.

Uploaded DataFrame:

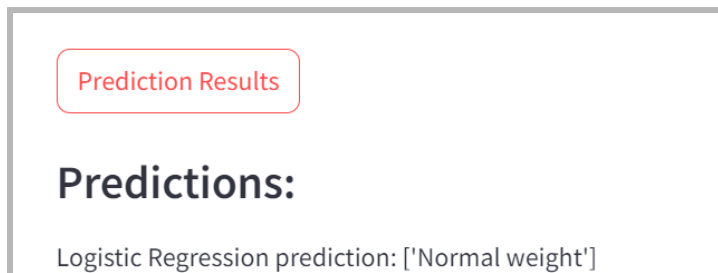
	Age	Gender	Height	Weight			
10	36	Male	179.0225	82.622	Type to search		
11	40	Female	149.8808	52.5184	23.3786	1	Normal weight
12	28	Male	180.1889	85.7793	26.4196	1	Overweight
13	28	Male	169.4988	55.3157	19.2537	1	Normal weight
14	41	Male	144.7066	82.1606	39.2362	1	Obese
15	70	Male	182.9818	78.0276	23.3041	1	Normal weight
16	53	Male	184.4417	82.2795	24.1865	2	Normal weight
17	57	Female	150.9549	51.9249	22.7867	3	Normal weight
18	41	Male	171.7542	78.8189	26.7187	1	Overweight
19	20	Female	183.8859	86.2839	25.5172	1	Overweight

The user is able to modify their metrics prior prediction.



The form is titled "Obesity Level Prediction Form" with a doctor icon. It contains several input fields: "Enter age" with a value of 0, "Select gender" with a dropdown menu showing "Male", "Enter height" with a value of 0.00, "Enter weight" with a value of 0.00, "Enter BMI" with a value of 0.00, and "Enter physical activity level" with a value of 1. Each input field has a minus and a plus button for adjustment.

A button is placed at the bottom to display the results of the prediction.



The results section features a red-outlined button labeled "Prediction Results". Below it, the heading "Predictions:" is followed by the text "Logistic Regression prediction: ['Normal weight']".

Testing by Making a Prediction

After the training of the model, testing was initiated. Using the fitness metrics of an active male 22 year-old college student, of height 180.22 cm, weight of 70.33kg, BMI of 22.22 and physical activity level of 4. A result classifying his Fitness/Obesity level was given, this individual was classified in the “Normal Weight” range.

```
✓ [177] # Making a new prediction after model was trained
0s X = np.array([[22,1,180.22,70.33,22.22,4]])
    X_new_scaled = scaler.transform(X)

✓ # Printing the prediction output
0s y_pred = model.predict(X_new_scaled)
    y_pred

    array(['Normal weight'], dtype=object)
```

The **decision support functionality** from the outcome of the prediction is that it will inform the customer on what fitness category they are in and, after gaining this knowledge, they can go over health plans and healthcare providers within the network of StriveLife company to select the best course of action based on the results and according to their needs. After the prototype is approved, future improvements like additional prompts could be added in the application for even greater results.

The prototype demo will not display any **security features**, but in order to use the data product once approved by the executives, the customer must create an account within StriveLife company guidelines, and after account creation, the user must log-in and navigate to the data product.

tools to monitor and maintain the product

In order to **monitor and maintain the product**, there will be a developer team in charge of taking care of any possible bugs that could be found in the tool, for example: invalid entry, constraints to the fitness metrics so that the customer does not add exaggerated inputs that could affect the functionality of the data product.

Maintenance will be done bi-weekly, and on-call developers will be available in case the data product page has any incidents. **User satisfaction surveys** will be prompted after the use of the data product for improvement and maintenance.

User-friendly, functional dashboard that includes three visualization types
Following two images use a dark-theme and display the GUI of the application.

The screenshot shows a web application titled "Fitness/Obesity Level Prediction Form" with a doctor icon. It features a "CSV File Uploader" section with a "Drag and drop file here" area (limit 200MB per file, CSV) and a "Browse files" button. Below this, a file named "obesity_data.csv" (75.7KB) is listed. The form includes input fields for "Enter age" (22), "Select gender" (Male), and "Enter height" (187.50).

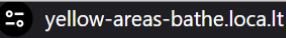
A file data uploader is located at the beginning of the app so the user testing the prototype can load the csv file required for execution of the prediction.

The screenshot shows the "Predictions:" section of the application. It displays the following information:

- Logistic Regression prediction: ['Normal weight']
- Random Forest prediction: ['Obese']
- Uploaded DataFrame:

	Age	Gender	Height	Weight	BMI	PhysicalActivityLevel	ObesityCategory
0	56	Male	173.5753	71.9821	23.8918	4	Normal weight
1	69	Male	164.1273	89.9593	33.3952	2	Obese
2	46	Female	168.0722	72.9306	25.8177	4	Overweight
3	32	Male	168.4596	84.8869	29.9122	3	Overweight

Continuing the GUI representation, in **light-theme**. There are three buttons at the bottom of the dataset display.



0	56	Male	173.5753	71.9821	23.8918	4	Normal weight
1	69	Male	164.1273	89.9593	33.3952	2	Obese
2	46	Female	168.0722	72.9306	25.8177	4	Overweight
3	32	Male	168.4596	84.8869	29.9122	3	Overweight
4	60	Male	183.5686	69.0389	20.4879	3	Normal weight
5	25	Female	166.4056	61.1459	22.0816	4	Normal weight
6	78	Male	183.5663	92.2085	27.3643	3	Overweight
7	38	Male	142.8751	59.3597	29.079	1	Overweight
8	56	Male	183.4786	75.1577	22.3256	4	Normal weight
9	75	Male	182.9741	81.5335	24.3532	2	Normal weight

Prediction Results

Show Accuracy of Results

Your category in relation to the rest of users:

The **‘Prediction Results’** button displays the results of the previous user input.

NOTE: Only the Logistic Regression prediction is displayed in order to keep a single output in the application. Accuracy still shows both trained models used in the prototype.

Prediction Results

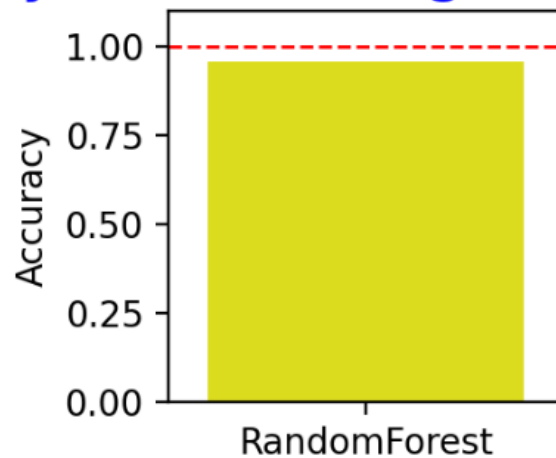
Predictions:

Logistic Regression prediction: ['Normal weight']

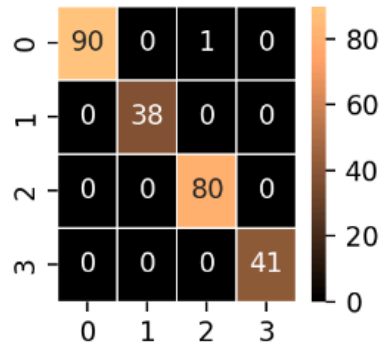
The **'Show Accuracy of Results'** button displays accuracy for both models trained for the prediction. These representations are explained in previous pages.

Show Accuracy of Results

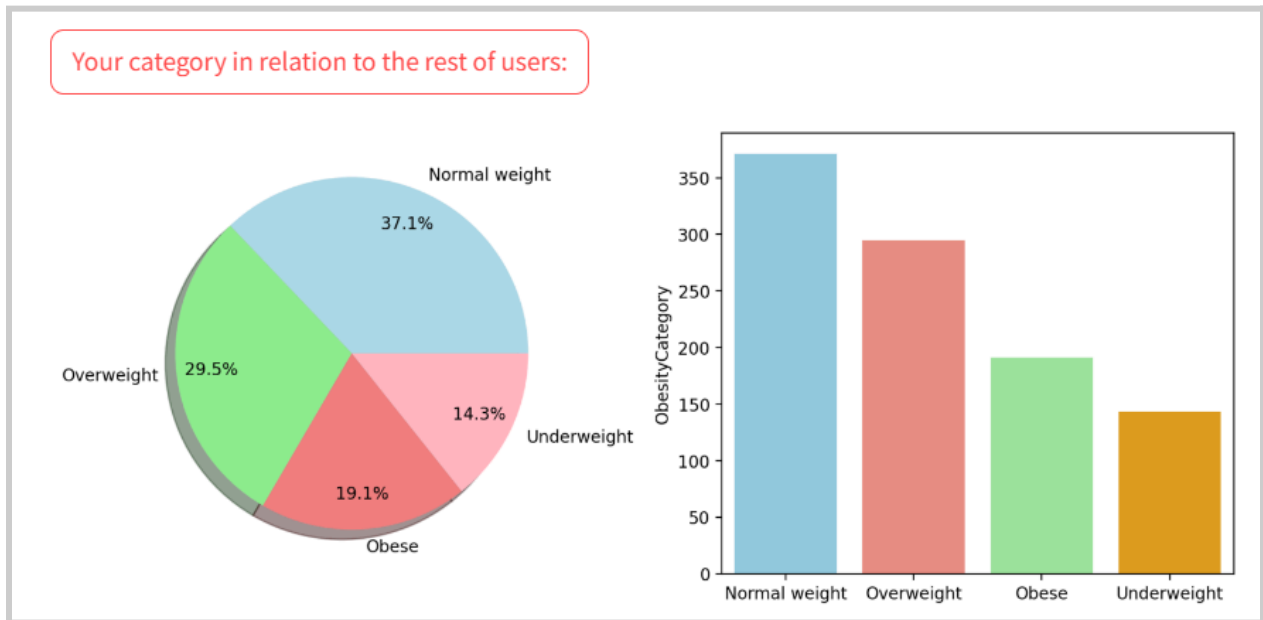
Accuracy Results: Logistic Regression



Accuracy Results: RandomForest Confusion Matrix



The **'Your category in relation to the rest of users:'** button, displays a graphical representation comparing the result to the rest of the individuals in the dataset.



Part D: Post-Implementation Report.

Solution Summary:

It is essential for the customers of StriveLife to have a tool that can let them know what their Obesity category is; as of today, in order for a customer to accurately choose one of our services, they must receive a physical check-up of their human body in order to determine what Fitness/Obesity category they fall in; and doing this involves taking their time to find a good medical practitioner that can have all the tools to accurately give the results that they need, once they receive such results they are finally able to choose one of our services.

The **implementation** of this data product will revolutionize and exchange the need of doing everything mentioned prior with inputting their own data, at the comfort of their home; while getting an accurate result. After results, they are prompted to sign up for services within the StriveLife company. The tool will also be helpful with making predictions of future body changes (future-weight loss for example), so that the customer can also see what Fitness/Obesity category level they could be able to fall in, if they reach the metrics that were input inside the future predictions.

This would make customers more likely to engage with our products and services, an estimate of 20-25% is calculated for increase in engagement with services after using the tool.

The **prototype was tested** with a dataset of a thousand individuals, and 500 individuals participated in the use of the prototype. The results were the following:

- Out of all 500 individuals = 496 people got true positive results
- The remaining 4 individuals got an error type 1 or 2. Either a false positive or negative.

This is overall an accuracy of 99.2%, which is outstanding and gives better expectations for the final product.

Additional results gathered from the 500 individuals who participated and 450 of them answered “yes” to a prompt asking for their interest in any health plans that StriveLife offers (90% of participants), **which accepts and validates the proposed hypothesis**. The tool can possibly attract more customers into StriveLife’s services.

If the company is able to generate more capital from this data product, all health and fitness plans can be updated according to any future predictions (this only if the user made a prediction of what their ideal fitness / obesity category level could be), and healthcare providers within the company network can also be informed about the data product; this way, they can advertise it to present patients, and the company would offer an incentive for those healthcare providers in exchange of traffic toward StriveLife’s services, a win-win.

Representation of raw and cleaned datasets with the code and executable files used to scrape and clean data.

In the following images, a representation of the **raw dataset** used in the prototype prior cleaning for easier data pre-processing. Along with the code used to pre-process said dataset. This is the **raw data**.

```
[26] setDatos = pd.read_csv('obesity_data.csv')
```

setDatos.head()

	Age	Gender	Height	Weight	BMI	PhysicalActivityLevel	ObesityCategory
0	56	Male	173.575262	71.982051	23.891783	4	Normal weight
1	69	Male	164.127306	89.959256	33.395209	2	Obese
2	46	Female	168.072202	72.930629	25.817737	4	Overweight
3	32	Male	168.459633	84.886912	29.912247	3	Overweight
4	60	Male	183.568568	69.038945	20.487903	3	Normal weight

The **LabelEncoder** method was used to transform the entire “Gender” column to non-numeric values, here is the cleaned dataset. The head method displays only the five initial rows. This is the **clean data**.

```
[34] le = LabelEncoder()
      setDatos["Gender"] = setDatos[["Gender"]].apply(le.fit_transform)
```

[35] setDatos.head()

	Age	Gender	Height	Weight	BMI	PhysicalActivityLevel	ObesityCategory
0	56	1	173.575262	71.982051	23.891783	4	Normal weight
1	69	1	164.127306	89.959256	33.395209	2	Obese
2	46	0	168.072202	72.930629	25.817737	4	Overweight
3	32	1	168.459633	84.886912	29.912247	3	Overweight
4	60	1	183.568568	69.038945	20.487903	3	Normal weight

The following code was used to represent the **analysis and descriptive representations** from previous pages in this document. **Boxplots and histograms**:

```
[30] plt.figure(figsize=(12,22))
      j = 1
      for i in columnNumerica:
          plt.subplot(5,2, j)
          sns.boxplot(setDatos[i], color="violet")
          plt.title(f"{i}")
          plt.subplot(5,2, j+1)
          sns.histplot(setDatos[i], kde=True, color="magenta", label=f"skewness:{setDatos[i].skew().round(2)}", bins=25)
          plt.title(f"{i}"), plt.xlabel("")
          j += 2
          plt.legend(fontsize="large")
      plt.tight_layout()
      plt.show()
```

The following code was used to represent the **heatmap**. Starting with the data cleansing, the columns for 'Gender' and 'ObesityCategory' were removed.

```
] noGenObese = setDatos.drop(["Gender", "ObesityCategory"], axis=1).corr()
noGenObese
```

	Age	Height	Weight	BMI	PhysicalActivityLevel
Age	1.000000	0.019647	-0.061400	-0.059971	-0.022308
Height	0.019647	1.000000	0.016033	-0.477091	0.030380
Weight	-0.061400	0.016033	1.000000	0.861438	0.063406
BMI	-0.059971	-0.477091	0.861438	1.000000	0.038020
PhysicalActivityLevel	-0.022308	0.030380	0.063406	0.038020	1.000000

After removal, the next code was used to represent the **heatmap**. Using the previous variable created, 'noGenObese', with the color palette 'rocket'.

```
) sns.heatmap(noGenObese, annot=True, lw=0.4, cmap="rocket")
plt.show()
```

The following code was used to represent the **pie charts and histograms** in relation to the Gender and ObesityCategory columns. The `value_counts()` method was used to return a series that contains counts of unique values for the Gender column. A for loop was used to iterate over the Gender and ObesityCategory columns. The plotting libraries (matplotlib and sns) are used to plot each type of graph.

```
▶ setDatos["Gender"].value_counts().index

plt.figure(figsize=(10,8))
j = 1
for i in ["Gender", "ObesityCategory"]:
    plt.subplot(2,2, j)
    sns.barplot(x=setDatos[i].value_counts().index, y=setDatos[i].value_counts())
    plt.xlabel("")
    plt.title(f"\n{i}")
    plt.subplot(2,2, j+1)
    plt.pie(x=setDatos[i].value_counts(), labels=setDatos[i].value_counts().index, autopct='%0.1f%%',
            pctdistance=0.8, shadow=True)
    j += 2
plt.tight_layout()
plt.show()
```


All source code can be found in the folder used for the submission of this prototype.

Here are some visuals from the source code used to represent the web application prototype for the data product.

Imports, functions to load models, take input and encoding Gender column,

```
paginaDePredict.py > train_and_plot
1 import streamlit as st
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 from sklearn.preprocessing import LabelEncoder
6 from sklearn.model_selection import train_test_split
7 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
8 from sklearn.preprocessing import StandardScaler
9 from sklearn.ensemble import RandomForestClassifier
10 from sklearn.linear_model import LogisticRegression
11 import seaborn as sns
12 import pickle
13 import warnings
14
15 warnings.filterwarnings("ignore", category=DeprecationWarning)
16
17
18 # Define function to load models from pickle files
19 def load_model(filename):
20     with open(filename, "rb") as f:
21         model = pickle.load(f)
22     return model
23
24 # Load models
25 logistic_model = load_model("LogisticRegression.pkl")
26 random_forest_model = load_model("RandomForestClassifier.pkl")
27
28 # Define function to take input for prediction
29 def take_input():
30     # Create input fields for user input
31     age = st.number_input('Enter age', min_value=0, step=1)
32     gender = st.selectbox('Select gender', ['Male', 'Female'])
33     height = st.number_input('Enter height', min_value=0.0)
34     weight = st.number_input('Enter weight', min_value=0.0)
35     bmi = st.number_input('Enter BMI', min_value=0.0)
36     physical_activity_level = st.number_input('Enter physical activity level', min_value=1, max_value=5, step=1)
37
38     # Convert gender to numeric
39     gender_numeric = 1 if gender == 'Male' else 0
40
41     # Return input as numpy array
42     return np.array([age, gender_numeric, height, weight, bmi, physical_activity_level])
```

Function to make predictions, and to train, split and plot data

```
paginaDePredict.py > take_input
44 #Define function to make predictions
45 def make_prediction(model, input_data):
46     return model.predict(input_data)
47
48
49 def train_and_plot(df):
50     le = LabelEncoder()
51     df["Gender"] = df[["Gender"]].apply(le.fit_transform)
52
53     y = df["ObesityCategory"]
54     X = df.drop("ObesityCategory", axis=1)
55
56     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
57
58     scaler = StandardScaler()
59     X_train_scaled = scaler.fit_transform(X_train)
60     X_test_scaled = scaler.transform(X_test)
61
62     import warnings
63     warnings.filterwarnings("ignore")
64
65     model_list = [RandomForestClassifier(), LogisticRegression()]
66
67     model_name_list = []
68     model_accuracies = []
69
70     for i in model_list:
71         model = i.fit(X_train_scaled, y_train)
72         model_name = model.__class__.__name__
73         y_pred = model.predict(X_test_scaled)
74         accuracy = accuracy_score(y_test, y_pred)
75
76         model_name_list.append(model_name)
77         model_accuracies.append(accuracy)
78
```

Function to display graphical representations inside the web app.

```
80 #Plotting accuracy graphs for both Models after
81 fig, ax = plt.subplots(figsize=(2,2))
82 cols = ["magenta" if i < (max(model_accuracies)) else "yellow" for i in model_accuracies]
83 sns.barplot(x=model_name_list, y=model_accuracies, ax=ax, palette=cols)
84 plt.ylim(0, 1.1)
85 plt.ylabel("Accuracy")
86 plt.axhline(1, lw=1, ls="--", color="red")
87 ax.set_xticklabels(["RandomForest", "Logistic"])
88 plt.title("Accuracy-Results: Logistic-Regression", fontsize=18, color="b")
89 st.pyplot(fig)
90
91 rf = RandomForestClassifier().fit(X_train_scaled, y_train)
92 y_pred_rf = rf.predict(X_test_scaled)
93
94 plt.figure(figsize=(2,2))
95 sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, linewidths=0.6, cmap="copper", fmt='.2g')
96 plt.title("Accuracy-Results: RandomForest-Confusion-Matrix", fontsize=18)
97 st.pyplot(plt)
98
99 #Plotting pie chart to see where user stands compared to the rest of dataset
100 def pie_chart(df):
101     plt.figure(figsize=(10,8))
102     j = 1
103     for i in ["ObesityCategory"]:
104         plt.subplot(2,2,j+1)
105         colors = ['skyblue', 'salmon', 'lightgreen', 'orange']
106         sns.barplot(x=df[i].value_counts().index, y=df[i].value_counts(), palette=colors)
107         plt.xlabel("")
108         plt.subplot(2,2,j)
109         colors = ['lightblue', 'lightgreen', 'lightcoral', 'lightpink']
110         plt.pie(x=df[i].value_counts(), labels=df[i].value_counts().index, autopct='%1.1f%%',
111               pctdistance=0.8, shadow=True, colors=colors)
112     j += 2
113     plt.tight_layout()
114     st.pyplot(plt)
```

The main method, including titles and subtitles for the web app, input taker, variables to make a prediction in a certain model, file uploader creator, and buttons for representation of results and graphics.

```
paginaDePredict.py > pie_chart
116 def main():
117     ...# Web app titles and sub titles
118     ...st.title('Fitness/Obesity Level Prediction Form :health_worker:')
119     ...st.header("Upload the test data prior execution.")
120     ...st.subheader("CSV File Uploader")
121
122     ...# File uploader
123     ...uploaded_file = st.file_uploader("Upload CSV file", type=["csv"])
124     ...
125     ...# Take input from user
126     ...input_data = take_input()
127
128     ...# Make predictions using the loaded models
129     ...logistic_prediction = make_prediction(logistic_model, input_data)
130     ...random_forest_prediction = make_prediction(random_forest_model, input_data)
131
132     ...# Display predictions
133     ...st.subheader('Predictions:')
134     ...st.write(f"Logistic Regression prediction: {logistic_prediction}")
135     ...st.write(f"Random Forest prediction: {random_forest_prediction}")
136
137     ...if uploaded_file is not None:
138     ...    ...# Read CSV file
139     ...    ...df = pd.read_csv(uploaded_file)
140     ...    ...
141     ...    ...# Display the DataFrame
142     ...    ...st.write("Uploaded DataFrame:")
143     ...    ...st.write(df)
144     ...
145     ...# Button to display prediction results
146     ...if st.button("Prediction Results"):
147     ...    ...# Display predictions
148     ...    ...st.subheader('Predictions:')
149     ...    ...st.write(f"Logistic Regression prediction: {logistic_prediction}")
150
151     ...# Button to display accuracy
152     ...if st.button("Show Accuracy of Results"):
153     ...    ...with warnings.catch_warnings():
154     ...    ...    warnings.simplefilter("ignore")
155     ...    ...    train_and_plot(df)
156
157     ...# Button to display pie chart
158     ...if st.button("Your category in relation to the rest of users"):
159     ...    ...with warnings.catch_warnings():
160     ...    ...    warnings.simplefilter("ignore")
161     ...    ...    pie_chart(df)
162
163 if __name__ == "__main__":
164     ...main()
```

How To Run Application

In order to run the application, the following must be done:

Open the following link

https://colab.research.google.com/drive/1vWO9PKhRPMvYyDUSeiTKKaTkVdr8oMyW?usp=drive_link

- Download the files attached to this submission. And open the 'txt' file for guidance.
- Once the files are downloaded, upload the notebook file to Google Colab.
- Once the notebook file is open, insert remaining files (py file and csv file) in the runtime directory of the notebook.
- When all files are set, run all existing notebook cells
- There is a specific cell that contains the following code:

```
import streamlit as st
! wget -q -O - ipv4.icanhazip.com
! streamlit run mainAppStreamKev.py & npx localtunnel --port 8501
```

With a result similar to the following:

```
34.120.240.63
Collecting usage statistics. To deactivate, set
browser.gatherUsageStats to False.
You can now view your Streamlit app in your browser.
Network URL: http://172.28.0.12:8501
External URL: http://34.23.88.242:8501
npx: installed 22 in 7.227s
your url is: https://wild-vans-pick.loca.lt
```

Copy the Tunnel Passcode that appears in the first line of the output, and open the link after 'your url is:', paste the address and press enter, the web application will be hosted.

You are about to visit: polite-jars-wave.loca.lt

This website is served for free via a [localtunnel](#).

You should only visit this website if you trust whoever sent this link to you.

Be careful about giving up personal or financial details such as passwords, credit cards, phone numbers, emails, etc. Phishing pages often look similar to pages of known banks, social networks, email portals or other trusted institutions in order to acquire personal information such as usernames, passwords or credit card details.

Please proceed with caution.

To access the website, please enter the tunnel password below.

If you don't know what it is, please ask whoever you got this link from.

Tunnel Password: *****

Click to Submit

Tunnel Password: 34.125.240.63

- Once the application is hosted, there will be a User Interface with the title of the application, a **file uploader** and some text boxes for user prompts. Each one of the text boxes represent fitness metrics. The user is required to type fitness metrics prior making a prediction.

The screenshot shows a web browser window with the address bar displaying 'four-socks-invite.loca.lt'. The page title is 'Fitness/Obesity Level Prediction Form' with a doctor icon. The main heading is 'Upload the test data prior execution.' followed by 'CSV File Uploader'. There is a file upload area with a cloud icon and text 'Drag and drop file here' and 'Limit 200MB per file • CSV'. A 'Browse files' button is present. Below this, a file named 'obesity_data.csv' (75.7KB) is shown with a close button. The form includes three input fields: 'Enter age' with a value of 22, 'Select gender' with a dropdown menu showing 'Male', and 'Enter height' with a value of 187.50.

- After all input typed, press enter and below all the input boxes, the prediction for Obesity Level/Fitness Level should be displayed for the user.

four-socks-invite.local.it

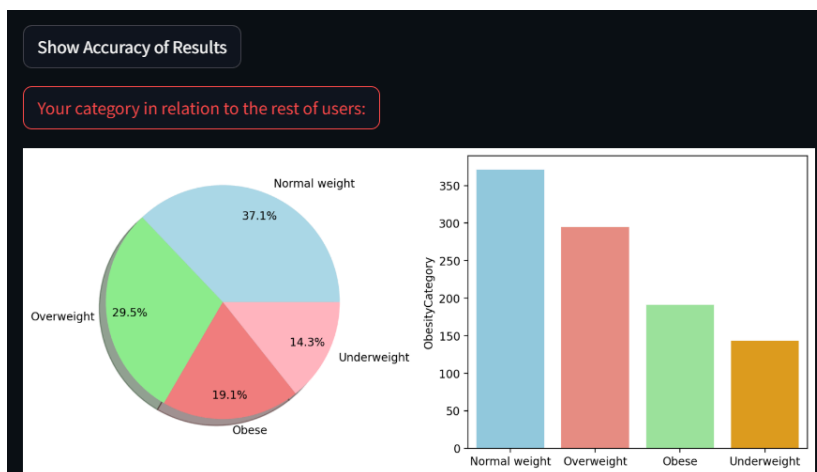
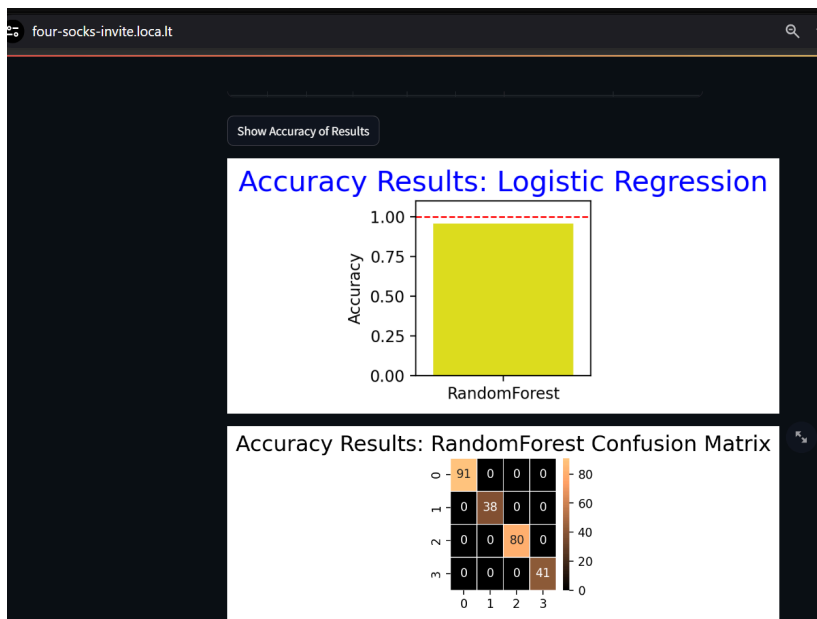
Enter weight
73.23

Enter BMI
23.22

Enter physical activity level
5

Predictions:
Logistic Regression prediction: ['Normal weight']

- Graphics will be displayed to show the accuracy and other measures in the app.



References

Box Plot (Box and Whiskers): How to Read One & How to Make One in Excel, TI-83, SPSS that takes references from:

Gonick, L. (1993). The Cartoon Guide to Statistics. HarperPerennial.

Kotz, S.; et al., eds. (2006), Encyclopedia of Statistical Sciences, Wiley.

Wheelan, C. (2014). Naked Statistics. W. W. Norton & Company

Salkind, N. (2016). Statistics for People Who (Think They) Hate Statistics: Using Microsoft Excel 4th Edition.

<https://www.statisticshowto.com/probability-and-statistics/descriptive-statistics/box-plot/>

What Is A Heatmap? By Mike Yi (2021)

<https://chartio.com/learn/charts/heatmap-complete-guide/>

Evaluating A Random Forest Model. JR Kreiger. (2020)

<https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56>

Sensitivity and Specificity. Wikipedia. Shelfskewed. (2024)

https://en.wikipedia.org/wiki/Sensitivity_and_specificity