Project Report on
COW DIET CLASSIFICATION

Submitted by
Tulasi Meghana Yalavarthi -R170941
Afzal Kamanuru-R170453

Under the guidance of
Shabana Shaik
Assistant Professor

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge and Technologies (RGUKT),R.K.Valley ,Kadapa ,Andhra Pradesh.

Rajiv Gandhi University of Knowledge Technologies

RK Valley, , Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled “COW DIET CLASSIFICATION” is a bonafied project work submitted by Tulasi Meghana Yalavarthi (R170941) and Afzal Kamanuru (R170453) in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer science and engineering for the year 2022-2023 carried out the work under the supervision of Ms.SHABANA.

Satyanandaram N	Ms.SK.Shabana
Head of CSE department	Project guide

DECLARATION

We, hereby declare that this report entitled “COW DIET CLASSIFICATION” submitted by us under the guidance and supervision of Sk.Shabana is a bonafide work . We also declare that it has not been submitted previously in part or in full to this university or other university or institution for the award of any degree or diploma. All information included from other sources have been duly acknowledged. We will be solely responsible if any kind of plagiarism is found.

Place : R.K.Valley

Y.Tulasi Meghana-R170941

K.Afzal-R170453

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our Director.Prof. K. SANDHYA RANI for fostering an excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr.N.SATYANANDARAM for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide at college Ms.SK.SHABANA for her guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

INDEX

1.ABSTRACT

This project aims to develop a predictive model to discriminate cow's diet, If they are grassfed or not using mid-infrared spectrometry. The development of this project could be of great help for the diary processors to guarantee the customers that the product origin from grass-fed animals. The dataset that will be used in this project is consisting of mir spectroscopy of milk .The data were analysed following different modelling strategies and the best model was selected based on cross-validation accuracy

Key words: Machine Learning, Classification, Ridge Classifier, Cross Validation

2.INTRODUCTION

Authentication of farming practices through rapid analyses of products has been a concern of many dairy production chains for several years. This has especially been a concern for European Protected Designation of Origin (PDO) cheeses, whose specifications require restrictions in farming practices related in particular to animal feeding. As such certifications often add value to products, producers need tools to check and strengthen the commitments made with consumers through labels and specifications. Animal feeding is considered one of the main factors affecting milk and other diary product's composition and sensory properties. Thus, several studies for the authentication of feeding systems on dairy products have been conducted in recent decades and recently reviewed by Prache etal (2020).Among others, Vicente et al. (2017), Hurtaud et al. (2014), and Mitani et al. (2016) successfully used milk fatty acids (FA) to authenticate cow feeding systems. Vitamins and carotenoids in milk were used to authenticate pasture feeding of cows (Segato et al., 2017), and volatile organic compounds were tested for authentication purposes (Valdivielso et al., 2017; Bergamaschi et al., 2020). Auerswald et al. (2015) and Valenti et al. (2017) proposed stable isotopes for cow feeding authentication. However, among the various authentication tools, Prache et al. (2020) identified infrared spectroscopy techniques as the most promising methods due to their rapidity and low cost, which allow for routine application, as well as for their accurate performances, which are often similar to those of reference methods used for authentication purposes.

The use of mid-infrared spectroscopy (MIRS) has become a relevant topic in agri-food sciences, due to its capacity to routinely quantify a wide range of important characteristics rapidly and costeffective. In particular, MIRS is nowadays commonly employed to monitor and quantify milk quality parameters, such as concentrations of fat, protein, casein, and lactose. These parameters are used for milk quality-based payment schemes, genetic and genomic selection, and as farmers' support tool. Spectral information generated from MIRS analysis have also proven to be effective in predicting fine milk quality parameters, including protein fractions, free amino acids individual and groups of fatty acids], milk processing traits. The aims of this work were to test the ability of MIR on bulk milk to predict indicators (1) of diet composition of dairy cows.

3.Purpose

The main purpose of the Cow Diet classification is that it takes the spectroscopic values of milk through the data set ,and helps us understand about the category of diet the cattle is fed among [Grass, TMR, Clover].From this farmers and consumers can easily get idea about the milk or the diary products origin .

4.Scope

The main objective of the project which is to develop a predictive model to discriminate cow’s diet.

- Collecting and pre-processing steps involve collecting milk spectroscopy data and cleaning the data, removing any outliers, and normalizing the data to ensure that it is suitable for use in machine learning algorithms.
- Feature extraction and selection: You will need to extract and select relevant features from the dataset that are most informative for predicting whether a cow's diet was grass-fed or not. This will involve applying mid-infrared spectrometry techniques and using domain knowledge to identify relevant features.
- Model development and evaluation: We develop and train machine learning models to predict whether a cow's diet was grass-fed or not using the extracted features. We then evaluate the performance of each model using crossvalidation techniques to select the best model.
- Implementation: Once you have identified the best performing model, we implement it and outputs a prediction of what the cow's diet was.
- **Applications**: Providing dairy processors with a tool to guarantee customers that their products originate from grassfed animals
- **Potential future work**: There may be potential for further research in this area. This could involve expanding the dataset to include more samples, exploring different modelling techniques or evaluating the impact of different pre-processing methods on model performance.

5.Requirement Specification

Hardware Configuration:

- Processor : A processor with a speed of 2.5Ghz or higher is sufficient for this project.
- Memory: A minimum of 8GB of RAM is recommended.
- Storage: 500GB of storage space is sufficient for this project.

Software Configuration:

Programming Language	Python
Packages	NumPy, Pandas, Scikit-Learn, TensorFlow,Matplotlib.
Operating System	Windows, Linux
Technology	Machine Learning

Python:

Python is a popular programming language for machine learning due to its simplicity, flexibility, and rich ecosystem of libraries and tools specifically designed for machine learning. Python has a large and active community of developers and users, which has contributed to the development of many powerful and user-friendly machine learning frameworks and libraries.

Libraries:

Some of the most popular machine learning libraries in Python include:

1. Scikit-learn: A machine learning library that provides simple and efficient tools for data mining and data analysis.

2. TensorFlow: A popular open-source machine learning framework developed by Google that is widely used for building and training deep learning models.

3. Matplotlib: It is a library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functionality, including line plots, scatter plots, bar charts, histograms, and more. Matplotlib is often used for visualizing data during the exploratory data analysis (EDA) phase of machine learning projects.

4. NumPy: It is a library that provides support for large, multi-dimensional arrays and matrices, as well as a variety of mathematical functions to manipulate them. It is a fundamental library for scientific computing in Python and is often used for tasks such as data manipulation, linear algebra, and statistical analysis.

5. Pandas: A library for data manipulation and analysis that provides powerful data structures for working with structured data.

In summary, Python is an excellent choice for developing machine learning models due to its simplicity, flexibility, and rich ecosystem of libraries and tools. It provides an easy-to-learn syntax and offers powerful and efficient libraries for building and training machine learning models.

Machine Learning:

Machine learning is a subfield of artificial intelligence that involves the development of algorithms and statistical models that enable computer systems to learn from and make predictions or decisions based on data. The goal of machine learning is to build systems that can automatically improve their performance on a given task over time, without being explicitly programmed.

The basic idea behind machine learning is to train a model using a set of input data and corresponding output data, known as a training set. The model then uses this training data to learn patterns or relationships in the data that can be used to make predictions or decisions on new, unseen data.

Machine learning has a wide range of applications, including image recognition, natural language processing, speech recognition, recommendation systems, fraud detection, and many others. Machine learning is becoming increasingly important in various industries, including healthcare, finance, transportation, and e-commerce.

6.Analysis and Design

To perform an analysis for cow diet Classification using machine learning, we need to first define the problem we want to solve. We want to authenticate the cow diet based on its milk spectroscopic values.

As we have defined the problem, we gather data on the cow's milk, and find the spectroscopic values of it .After collecting the data, we need to pre-process it, which involves cleaning and transforming the data so that it is suitable for machine learning. This includes dealing with cleaning the data, handling outliers.

Next, we need to choose an appropriate machine learning algorithm for the problem at hand, different classifier models. We also need to tune the algorithm's parameters to optimize its performance.

We then train the model on the pre-processed data, using a portion of the dataset for training and another portion for validation. We evaluate the model's performance on the validation dataset to ensure that it can accurately predict the diets of cows in test data.

Design:

The most creative and challenging phase of the life cycle is system design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in implementations of the candidate system. The design may be defined as "the process of applying various techniques and principles for the purpose of defining a device, a process or a system with sufficient details to permit its physical realization."

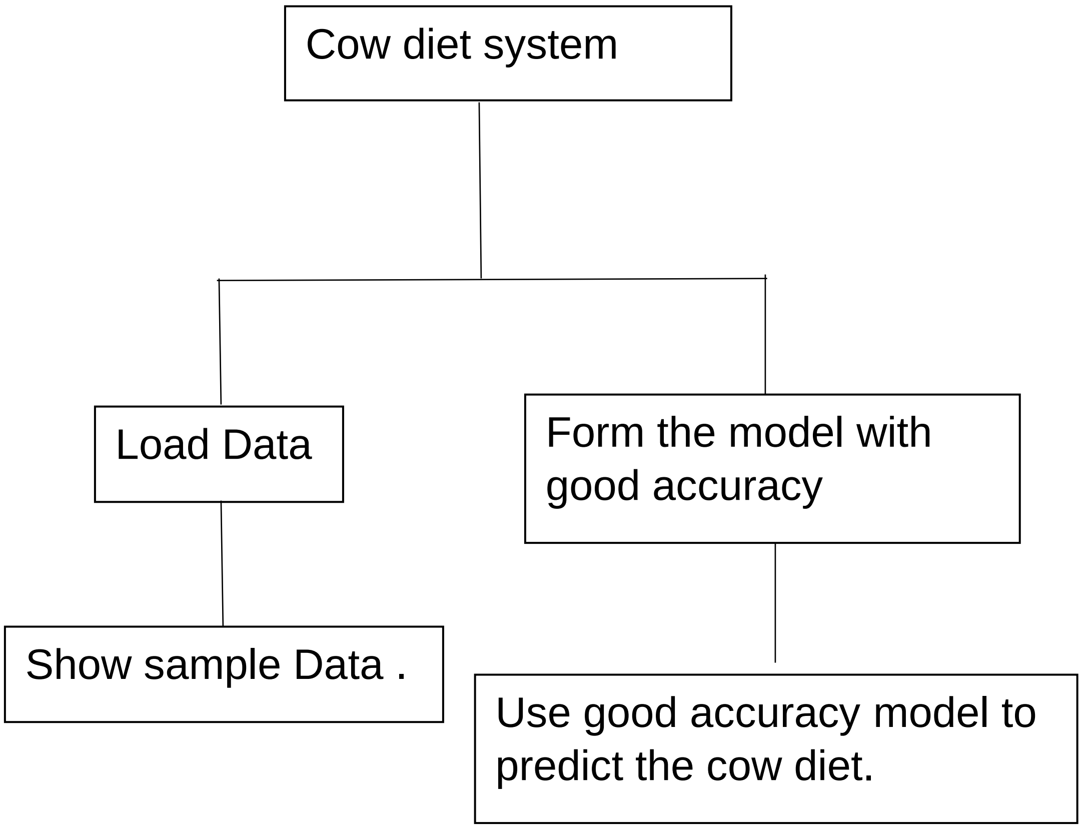
UML Diagrams:

The Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing and documenting of a software intensive system. The UML gives a standard way to write a system blueprints, covering conceptual things, such as classes written in a specified programmed language, database schemas and reusable software components.

- Use-Case Diagram
- Activity Diagram

Use case Diagram:

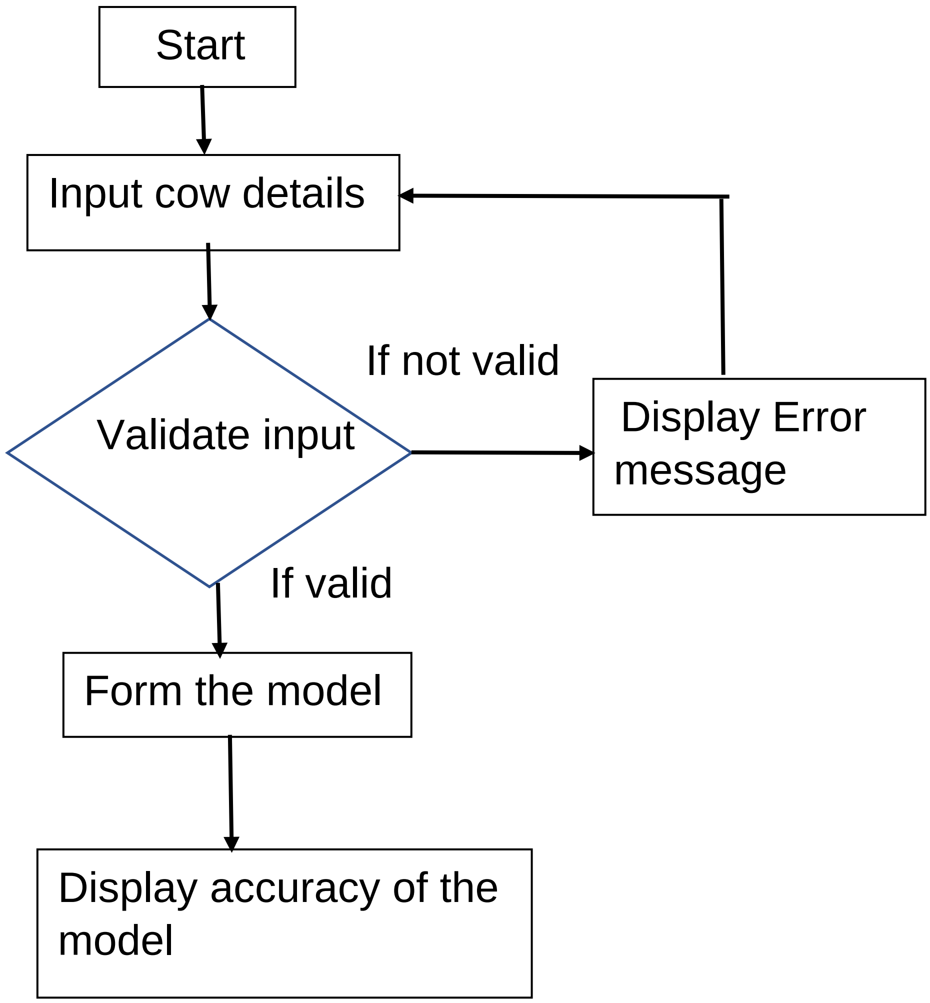
The use case diagram shows the primary actors and their interactions with the Cow Diet System. The two main use cases are loading the data and predicting the cow's diet. After loading the data, the dataset can be displayed to the user. Once the cow's diet is predicted system displays the prediction to the user.



Activity Diagram:

An activity diagram is a type of UML (Unified Modeling Language) diagram that models the o flow of activities or actions within a system or process. Activity diagrams are commonly used in software design and business process modeling to describe the steps and interactions involved in a particular process or use case.

The basic building blocks of an activity diagram are actions, which represent specific steps or activities within the process being modeled, and transitions, which show the flow of control between the actions. Additional elements such as decisions, forks, and joins can be used to model more complex processes



7.Implementation and Testing

After the model have been perfectly created with good accuracy, the system will be implemented and the system can be used.

System Testing

The goal of the system testing process was to determine all faults in our project .The program was subjected to a set of test inputs and many explanations were made and based on these explanations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing

1. Unit testing

2 .Integration testing <u>Unit Testing:</u>

Unit testing involves testing individual units or components of the software in isolation to ensure they function correctly. This could involve testing individual functions or methods that calculate nutritional values or recommend diets for specific cows. Unit testing can be automated using testing frameworks and can help catch bugs early in the development process.

Unit testing could be used to test individual functions or algorithms that help form models or classify cow diets for specific milk data.

<u>Integration Testing:</u>

Integration testing involves testing the entire system as a whole to ensure that all of the components work together correctly. This could involve testing the integration of different modules, the accuracy of the predictions made by the software, and the overall user experience of the system. Integration testing is done manually and can help us identify any issues that may arise when different components of the software are combined.

In the context of the cow diet prediction project Integration testing could be used to test the overall accuracy of the predictions made by the software and to ensure that the user interface is intuitive and easy to use. Both types of testing are important for ensuring the quality and reliability of the software.

Sample Code:

Dataset

Data used in this study originated from Teagasc Moorepark Dairy Research Farm (Fermoy, Co. Cork, Ireland) between May and August in 2015, 2016, and 2017. A total of 120 HolsteinFriesian cows from different parities were involved in the experiment across the years, with a mean number of 36 samples per cow, and with some of the cows participating in the experiment in more than 1 yr. Each year, 54 cows were randomly assigned to different dietary treatments for the entire lactation period. The treatment diets included grass (GRS), which consisted of cows maintained outdoors on a perennial ryegrass sward only, clover (CLV), where cows were maintained outdoors on a perennial ryegrass white clover sward (with an annual average clover content of 20%) only, and TMR, where cows were maintained indoors and fed with a single nutritional mix containing grass silage, maize silage, and concentrates. Further information on the experimental design and dietary treatments have been described by O'Callaghan et al. (2016). The cows were milked twice daily (0730 and 1530 h), and a.m. and p.m. milk samples were collected once weekly from consecutive milkings and analysed by a Pro-Foss FT6000 (FOSS). A total of 4,364 milk spectra were stored, comprising 1,060 wavelengths in the region from 925 cm^{-1} and 5,010 cm^{-1}. The wavelengths values were recorded as transmittance values.

This is a classification problem where 3 classes are targeted:

- GRS
- CLV
- TMR

Data are sampled from 54 cows across 3 years. Each data points contain 1060 wavelengths within a specified light region.

Code

IMPORTING LIBRARIES:

```
import sys import importlib
from pathlib import Path
import numpy as np import
pandas as pd import
matplotlib.pyplot as plt
sys.path.insert(0, '..') from
src import utils from
sklearn.feature_selection
import SelectFromModel
from sklearn.neighbors
import KNeighborsClassifier
from sklearn.linear_model
import RidgeClassifierCV
from sklearn.tree import
DecisionTreeClassifier from
sklearn.ensemble import
RandomForestClassifier
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,
QuadraticDiscriminantAnalysis from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split from sklearn
import metrics from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score from
sklearn.model_selection import cross_val_predict from
sklearn.model_selection import StratifiedKFold from
sklearn.preprocessing import StandardScaler from sklearn.metrics
import accuracy_score from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder # Reading the
datasets into dataframes
full_data = Path('..', 'data', 'full_dataset.xlsx')
train_path = Path('..', 'data', 'raw_train.csv') test_path
= Path('..', 'data', 'raw_test.csv')

train_df = utils.read_raw_data(full_data, train_path, 0) test_df
= utils.read_raw_data(full_data, test_path, 1)
```

read_raw_data function:

```
def read_raw_data(full_path: Path, dest_path: Path, sheet: int) -> pd.DataFrame:
    df = None

    if dest_path.exists():
        df = pd.read_csv(dest_path)
else:
        df = pd.read_excel(full_path, sheet, engine='openpyxl')
df.to_csv(dest_path, index=False)
```

```
    return df
```

 # Fixing the dataset

```
train_df.drop(train_df[train_df.col1<1].index.tolist(), axis=0, inplace=True)
test_df.drop(test_df[test_df.col1<1].index.tolist(), axis=0, inplace=True)
train_df.reset_index(inplace=True, drop=True) test_df.reset_index(inplace=True,
drop=True)
```

Remove outlier spectra with col1 < 1

```
train_df = train_df[train_df['col1'] >= 1]
train_df['Diet'].value_counts() test_df
= test_df[test_df['col1'] >= 1]
test_df.shape
```

#Plot to show medians of TMR ,GRS AND CLV

```
clv = train_df[train_df['Diet'] == 'CLV'].drop(['Diet'], axis=1) grs =
train_df[train_df['Diet'] == 'GRS'].drop(['Diet'], axis=1) tmr =
train_df[train_df['Diet'] == 'TMR'].drop(['Diet'], axis=1) fig =
plt.figure(figsize=(50, 15)) plt.plot(np.median(tmr.values, axis=0),
label='TMR Median', color='red') plt.plot(np.median(grs.values, axis=0),
label='GRS Median', color='green') plt.plot(np.median(clv.values,
axis=0), label='CLV Median', color='blue') plt.vlines(172, ymin=0,
ymax=2, color='red', linestyle='--') plt.vlines(205, ymin=0, ymax=2,
color='red', linestyle='--') plt.vlines(536, ymin=0, ymax=2, color='red',
linestyle='--') plt.vlines(728, ymin=0, ymax=2, color='red', linestyle='--')
plt.vlines(748, ymin=0, ymax=2, color='red', linestyle='--')
plt.vlines(1059, ymin=0, ymax=2, color='red', linestyle='--') plt.legend()
plt.show()
```

#Target variable: Diet column

```
target = train_df['Diet'] target_columns = target.unique()
features = [col for col in train_df.columns if col not in 'Diet']
target_stats = {}

for target_value in target_columns:
target_stats[target_value] = {
        'count': train_df[train_df['Diet'] == target_value].shape[0],
        'perc': train_df[train_df['Diet'] == target_value].shape[0] / train_df.shape[0]
    }
print(f'Unique values in the target column: {target_columns}')
print() for t, v in target_stats.items():
    count = v['count']     perc = v['perc']     print(f'Target
{t}: {count} instances ({perc}% of total)')
```

 # predict_eval function to show confusion matrix and accuracy

```
def predict_eval(clf, y, y_pred):

    actual_vs_predicted = pd.concat([y, pd.DataFrame(y_pred, columns=['Predicted'],
index=y.index)], axis=1)     actual_vs_predicted.head()     print(f'Accuracy:
{accuracy_score(y, y_pred)}')

    cm = confusion_matrix(y, y_pred, labels=unique_labels(y))
disp       =       ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=unique_labels(y))                          disp.plot()
plt.grid(False)         plt.show()
```

training and test data variables:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
```

```
print("Training data shape: ", X_train.shape, y_train.shape)
print("Test data shape: ", X_test.shape, y_test.shape)
```

choosing algorithms with accuracies:

```
list_clf = [
        #RidgeClassifier(),
        #RidgeClassifierCV(),
        #RidgeClassifierCV(normalize=True),
        RidgeClassifierCV(alphas=np.logspace(-3, 3, 10)),
        #RidgeClassifierCV(alphas=np.logspace(-4, 4, 10)),
        RidgeClassifierCV(alphas=np.logspace(-5, 5, 10)),
        RidgeClassifierCV(alphas=np.logspace(-6, 6, 10)),
#RidgeClassifierCV(alphas=np.logspace(-10, 10, 10), normalize=True),
        #LogisticRegression(penalty='l2'), LogisticRegression(penalty='l1'),
        #GaussianNB(),
        LinearDiscriminantAnalysis(),
        #QuadraticDiscriminantAnalysis(), LinearSVC(),
        #RandomForestClassifier(n_estimators=100),
        #GradientBoostingClassifier(n_estimators=100),
        #ExtraTreesClassifier(n_estimators=100)
        #RotationForestClassifier(n_estimators=25, n_features_per_subset=3,
rotation_algo='randomized'),
        #KNeighborsClassifier(n_neighbors=3),
        #MLPClassifier(alpha=1, max_iter=1000),
        #SVC(kernel="linear", C=0.025),
        #LinearSVC(C=0.001),
        #SVC(C=500.0, kernel="poly", degree=2, coef0=0, gamma=1.0)
        #SVC(gamma=2, C=1)
        ]
```

```
def algoCV(list_clf, X, y, folds = 3, error_measures = ['accuracy']):

    algos_df = pd.DataFrame({"algo":[], "acc":[]})

    for algo in list_clf:
        print(algo)        scores = cross_validate(algo, X, y,
scoring=error_measures, cv=folds)
        #scores        acc =
scores['test_accuracy']        fit_time =
np.mean(scores['fit_time'])
score_time =
np.mean(scores['score_time'])
algos_df_new_row =
pd.DataFrame({"algo":[str(algo)],
"acc":[np.mean(acc)]})        algos_df =
pd.concat([algos_df,algos_df_new_row])

        print("acc: ", acc)
print("accCV: ", np.mean(acc))
#print("fit_time: ", fit_time)
        #print("score_time: ", score_time)
        #print("time fit+score:", fit_time + score_time)
print("\n")
     print(algos_df.sort_values('acc', ascending=False))
algos_df.sort_values('acc').to_csv("cv" + str(folds) + "-acc.csv")

if __name__ == "__main__":
```

```
   algoCV(list_clf, X, y)
```

Implementing feature selection and produce accuracy with the confusion matrix:

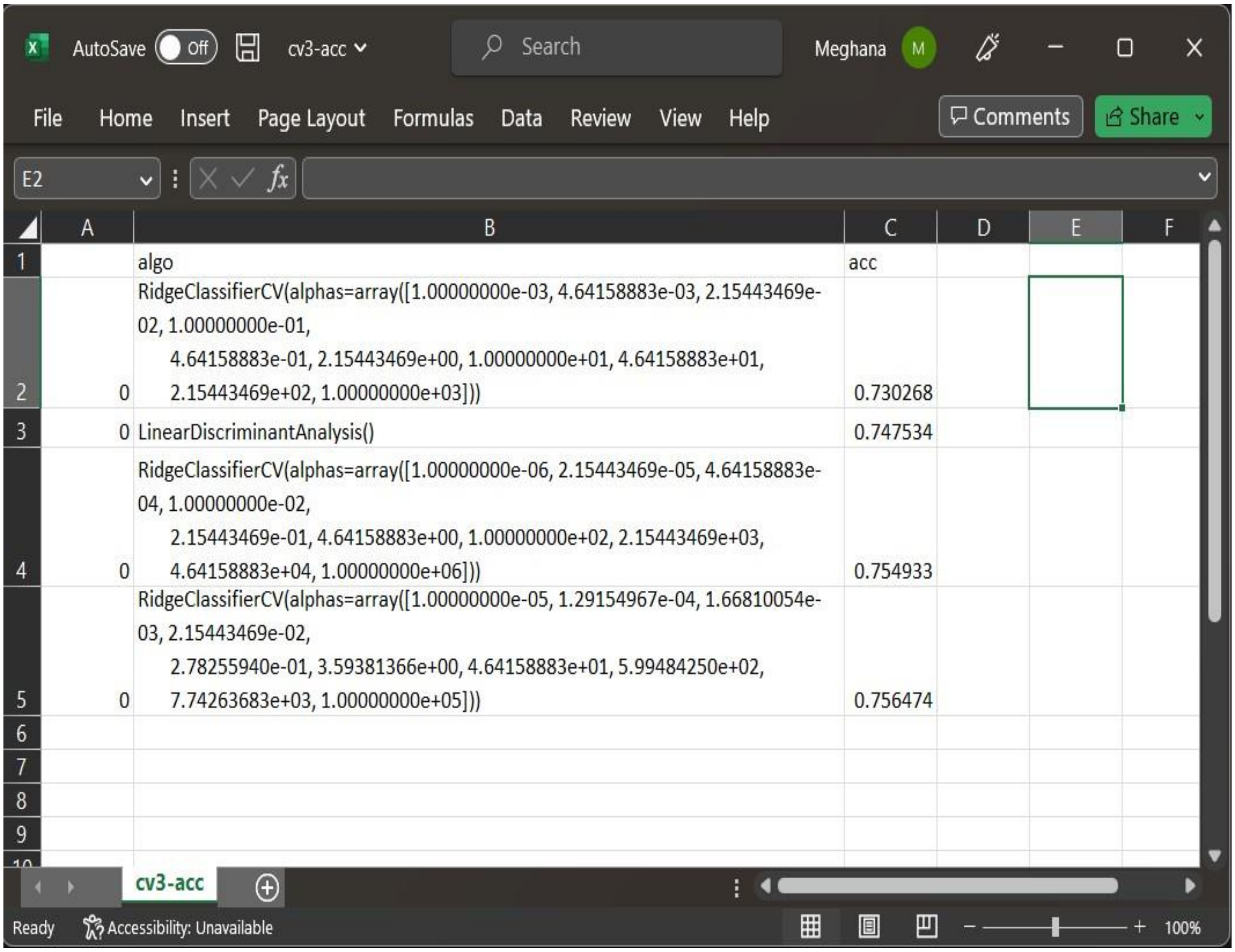
```
clf = Pipeline([
   ('feature_selection', SelectFromModel(RidgeClassifierCV(alphas=np.logspace(-5, 5,
10)))),


  ('classification', RidgeClassifierCV(alphas=np.logspace(-5, 5, 10)))
])
clf.fit(X_train, y_train) y_pred
= clf.predict(X_test)
df=pd.DataFrame(y_pred)
df.to_csv('file1.csv') predict_eval(clf,
y_test, y_pred)
```

Evaluation:

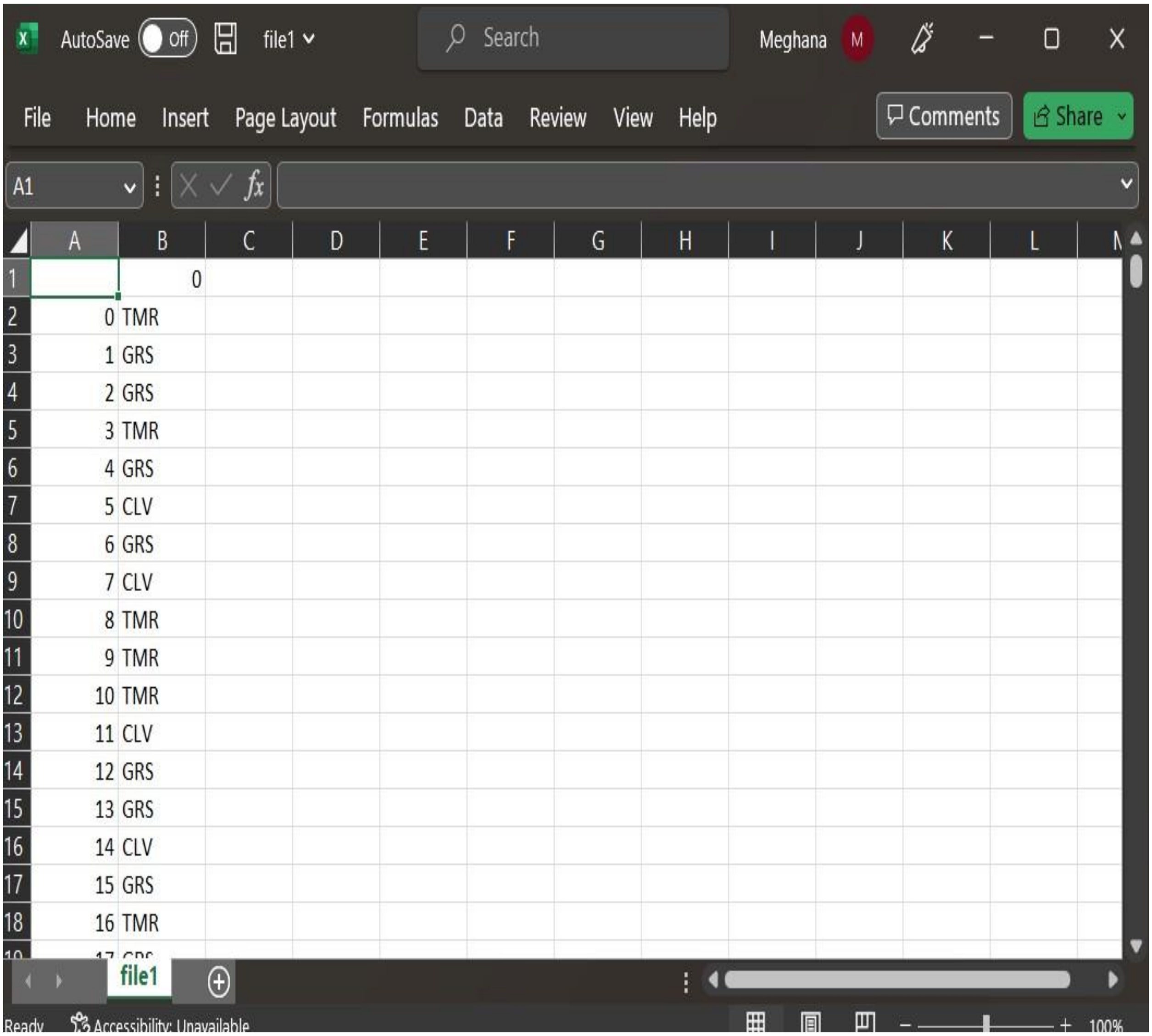
The evaluation creates csv files to show us the accuracy of different models that we choose to implement and the predicted values in two separated files .Then in the end it shows the accuracy of the model and gives us the confusion matrix of the model with highest accuracy.

CSV FILE OF ACCURACIES OF DIFFERENT MODELS:

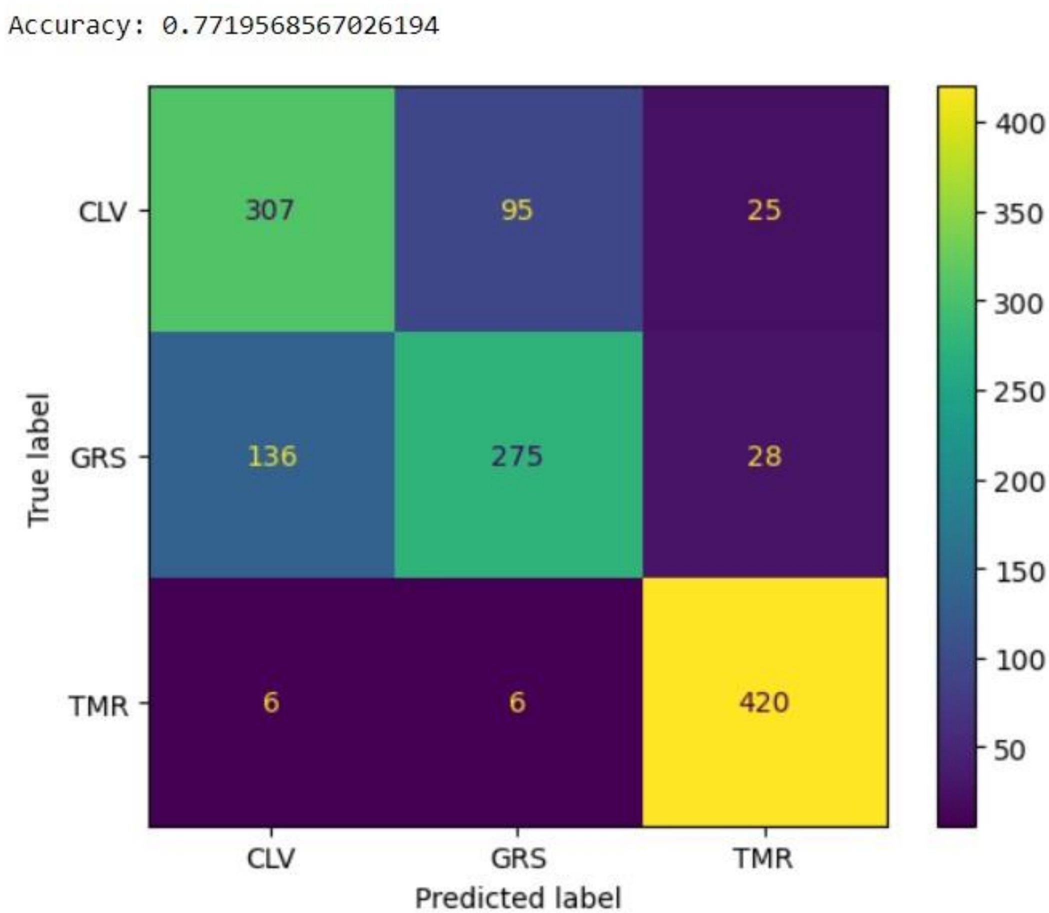


	A	B	C
1		algo	acc
2	0	RidgeClassifierCV(alphas=array([1.00000000e-03, 4.64158883e-03, 2.15443469e-02, 1.00000000e-01, 4.64158883e-01, 2.15443469e+00, 1.00000000e+01, 4.64158883e+01, 2.15443469e+02, 1.00000000e+03]))	0.730268
3	0	LinearDiscriminantAnalysis()	0.747534
4	0	RidgeClassifierCV(alphas=array([1.00000000e-06, 2.15443469e-05, 4.64158883e-04, 1.00000000e-02, 2.15443469e-01, 4.64158883e+00, 1.00000000e+02, 2.15443469e+03, 4.64158883e+04, 1.00000000e+06]))	0.754933
5	0	RidgeClassifierCV(alphas=array([1.00000000e-05, 1.29154967e-04, 1.66810054e-03, 2.15443469e-02, 2.78255940e-01, 3.59381366e+00, 4.64158883e+01, 5.99484250e+02, 7.74263683e+03, 1.00000000e+05]))	0.756474

CSV FILE OF PREDICTED VALUES:



ACCURACY AND CONFUSION MATRIX:



Conclusion:

In conclusion, cow diet prediction using machine learning is a complex task that involves several steps, including problem definition, data collection, preprocessing, algorithm selection, model training, evaluation. It is worth noting that classification models based on spectroscopic analysis of milk have been used to distinguish between different feeding regimes in cows, including grass-fed and non-grass-fed diets. These methods can be useful for verifying labeling claims, ensuring product quality, and informing consumers about the products they purchase.

Overall, cow diet prediction using machine learning has the potential to revolutionize the way we feed and manage cows, leading to increased efficiency and productivity in the dairy industry.

REFRENCES:

- Data challenge at the [International Workshop on Spectroscopy and Chemometrics 2022]

(https://www.vistamilk.ie/event/international-workshop-onspectroscopy-chemometrics/), organised by the Vistamilk SFI Research Centre, Ireland.

- Article on Cow Diet Prediction is published on science direct

https://www.sciencedirect.com/science/article/pii/S0022030220308912

- An arXiv preprint published on https://arxiv.org/abs/2210.04479

**********************THANKYOU**********************