

Project Title: "TasteCraft : Enhancing Food Flavor Profiles through Data-Driven Innovation in the Food and Beverage Industry"

By Pavithra. M

II nd year CSBS – "A"

Data Science Intern at Brainovision Pvt .Ltd

Rajalakshmi Engineering College

TABLE OF CONTENTS		
S.NO	TOPIC	PAGE.NO
1.	Problem Statement	03
2.	Abstract	04
3.	Data Collection	05
4.	Feature Engineering	09
5.	Dataset Creation (Sample Data)	12
6.	Sample Data to Population Data Creation	13
7.	Modeling Linear Regression	14
8.	Support Vector Machine (Regression)	16
10.	Random Forest (Regression)	20
11.	Model Evaluation	22
12.	Model Comparison	24
13.	Discussions	26
14.	Conclusion	28
15.	References	29

Problem Statement :

In the Food and Beverage industry's Research and Development department, a critical challenge is to gain a comprehensive understanding of the factors influencing the taste of food. The aim is to enhance product development strategies and elevate customer satisfaction. Leveraging data science techniques, specifically employing machine learning models such as linear regression and support vector machine for regression, the goal is to decipher the intricate relationships within the dataset. The dataset includes features like Flavor Intensity, Texture Complexity, Spice Profile, Freshness Index, Culinary Technique Score, and Aroma Richness, with tastiness as the target variable. The envisioned solution involves a feature engineering process to extract meaningful insights from the dataset, facilitating the development of accurate models. This approach aims to provide actionable recommendations for targeted improvements in product formulation and flavor profiles.

Industry: Food and Beverage**Department:** Research and Development**Issue:** Understanding the factors influencing the taste of food to enhance product development and customer satisfaction.**Solution:** Utilize data science techniques to analyze sensory data, ingredient compositions, and consumer feedback to identify key variables affecting taste, leading to targeted improvements in product formulation and flavor profiles.

Abstract:

In the dynamic landscape of the Food and Beverage industry, the Research and Development department faces a pivotal challenge – comprehending the intricate factors that shape the taste of food products. This abstract outlines a strategic solution employing advanced data science techniques to dissect sensory data, ingredient compositions, and valuable consumer feedback. The objective is to unravel the nuances of taste influencers and distill actionable insights for targeted improvements in product formulation and flavor profiles. By integrating data-driven methodologies, this initiative seeks to elevate product quality, amplify consumer satisfaction, and fortify the competitive edge of the organization in the ever-evolving market.

DATA COLLECTION:

For the above Problem statement the data is collected across various web resources available in Google by using the processes involved in feature engineering.

FEATURES/ATTRIBUTES:

1. Ingredient composition
2. Cooking time and temperature
3. Salt content
4. Sweetness level
5. Umami intensity
6. Spices and herbs
7. Texture and mouthfeel
8. Fat content
9. Acidity level
10. Cooking method
11. Presentation and aesthetics
12. Serving temperature
13. Food freshness
14. Cultural influences
15. Dietary restrictions
16. Processing techniques
17. Seasonal variations
18. Proportion of ingredients
19. Flavor balancing
20. Food pairing
21. Smell and aroma
22. Cooking oil used
23. Food origin and source
24. Food additives
25. Cooking utensils/materials
26. Water quality
27. pH level
28. Cooking environment
29. Personal preferences

30. Food appearance and color

FEATURE/ATTRIBUTE/DATA DESCRIPTION:

1. **Ingredient composition:** The combination and proportion of various food components, such as proteins, carbohydrates, and fats, that contribute to the overall dish.
2. **Cooking time and temperature:** The duration and heat level applied during the cooking process, influencing texture, flavor, and doneness of the food.
3. **Salt content:** The amount of salt used, impacting flavor enhancement and overall taste perception.
4. **Sweetness level:** The degree of sweetness present in the dish, affecting the overall flavor profile and palatability.
5. **Umami intensity:** The strength of the savory and rich taste associated with umami, often derived from ingredients like mushrooms, soy sauce, or certain meats.
6. **Spices and herbs:** The selection and quantity of aromatic substances used to enhance the flavor and aroma of the dish.
7. **Texture and mouthfeel:** The tactile sensations experienced during eating, including factors like tenderness, crispiness, or creaminess.
8. **Fat content:** The amount of fats, both visible and hidden, contributing to flavor, mouthfeel, and overall richness of the dish.
9. **Acidity level:** The presence and degree of acidity, influencing the perceived freshness and balance of flavors.
10. **Cooking method:** The technique employed in preparing the dish, such as roasting, frying, steaming, or grilling, affecting texture and flavor development.
11. **Presentation and aesthetics:** The visual appeal and arrangement of the dish, influencing the overall dining experience.
12. **Serving temperature:** The temperature at which the dish is served, impacting the perception of flavors and textures.

13. **Food freshness:** The quality of ingredients in terms of being recently harvested or prepared, influencing taste and nutritional value.
14. **Cultural influences:** The culinary traditions, practices, and regional influences that shape the preparation and flavors of the dish.
15. **Dietary restrictions:** Any specific limitations or requirements related to dietary preferences, allergies, or health considerations.
16. **Processing techniques:** The methods used to transform raw ingredients into the final product, affecting flavor, texture, and nutritional content.
17. **Seasonal variations:** The availability and use of ingredients based on the season, impacting flavor and sustainability.
18. **Proportion of ingredients:** The balance and ratio of different components in the dish, ensuring a harmonious flavor profile.
19. **Flavor balancing:** The art of adjusting and harmonizing various taste elements like sweet, salty, sour, bitter, and umami.
20. **Food pairing:** The strategic combination of foods to enhance and complement each other's flavors.
21. **Smell and aroma:** The distinctive scents arising from the dish, contributing to the overall sensory experience.
22. **Cooking oil used:** The type and quantity of oil employed in cooking, influencing flavor and health considerations.
23. **Food origin and source:** The geographical and source-related aspects of ingredients, impacting taste and ethical considerations.
24. **Food additives:** Any additional substances used in food preparation for preservation, color, or flavor enhancement.
25. **Cooking utensils/materials:** The tools and materials utilized in cooking, affecting the cooking process and potentially imparting subtle flavors.
26. **Water quality:** The purity and characteristics of water used in cooking, influencing the taste and preparation of certain dishes.
27. **pH level:** The acidity or alkalinity of the dish, contributing to flavor balance and stability.

28. **Cooking environment:** The conditions under which the dish is prepared, including factors like humidity and altitude, affecting cooking outcomes.
29. **Personal preferences:** Individual taste and liking, influencing choices regarding seasoning, spice levels, and overall flavor profiles.
30. **Food appearance and color:** The visual characteristics of the dish, influencing the perceived freshness and desirability.

FEATURE ENGINEERING:

Feature engineering is a crucial aspect of the machine learning and data analysis process that involves creating new features or modifying existing ones to improve the performance of a model in solving a particular task. The goal of feature engineering is to enhance the representation of the data, making it more suitable for the algorithms used in a given machine learning problem.

Let's perform feature engineering to extract a subset of key features that could significantly influence the taste of food:

1. **Flavor Intensity:** A combination of sweetness, saltiness, and umami intensity.
2. **Texture Complexity:** Reflecting the diversity and complexity of textures in the dish.
3. **Spice Profile:** A representation of the types and quantities of spices used.
4. **Freshness Index:** Incorporating factors like ingredient freshness and overall dish freshness.
5. **Culinary Technique Score:** Evaluating the impact of cooking methods on taste.
6. **Aroma Richness:** Measuring the richness and complexity of the dish's aroma.

These six features capture essential aspects of taste, texture, freshness, culinary techniques, and aroma, providing a more focused set for analysis.

FEATURE ENGINEERING PROCESS EXPLANATION:

In the performed feature engineering process, we aimed to distill the complex nature of food taste into a more concise set of six features. Here's an explanation of each engineered feature:

1. **Flavor Intensity:** This feature combines sweetness, saltiness, and umami intensity to provide a comprehensive measure of the overall flavor strength in a dish.
2. **Texture Complexity:** Reflects the diversity and complexity of textures present in the food, capturing the interplay between various textures and their impact on taste.
3. **Spice Profile:** Represents the types and quantities of spices used in the dish, encapsulating the unique flavor characteristics contributed by the spices.
4. **Freshness Index:** Incorporates factors related to ingredient freshness and overall dish freshness, acknowledging the importance of using fresh ingredients in enhancing taste.
5. **Culinary Technique Score:** Evaluates the impact of various cooking methods on the taste, recognizing the influence of techniques such as grilling, roasting, or steaming.
6. **Aroma Richness:** Measures the richness and complexity of the dish's aroma, capturing the aromatic elements that significantly contribute to the overall taste experience.

By focusing on these six features, we aim to simplify the representation of taste while retaining key aspects related to flavor, texture, freshness, cooking techniques, spices, and aroma. This streamlined set of features provides a more manageable and targeted approach for further analysis in the context of food taste.

FEATURE ENCODING:

Here's a mapping of the numeric representations to the original categorical values:

• Spice Profile:	
	• 0: Mild
	• 1: Medium
	• 2: Balanced

	<ul style="list-style-type: none">• 3: Spicy
• Aroma Richness:	<ul style="list-style-type: none">• 0: Low• 1: High• 2: Moderate• 3: Very High

These mappings represent the original categorical values corresponding to the numeric representations in the dataset.

DATA PREPROCESSING:

The below dataset does not require any preprocessing techniques as we have already done feature encoding in the feature engineering process itself (Categorical value to numerical value conversion).

DATASET CREATION:

A manual sample dataset created in excel as a Yummy.csv file using the extracted six features as columns and including observations of 50 rows where the target variable is the tastiness column which is obtained by making mathematical calculations on the other attributes present.

SAMPLE DATA (Yummy.csv):

Yummy.csv							Open with ▾
A	B	C	D	E	F	G	
Flavor Intensity	Texture Complexity	Spice Profile[0:Mild	Freshness Index	Culinary Technique	Aroma Richness[0:L	Tastiness	
8.5	4.2	2	9	7.8	1	0.792682927	
7.2	3.5	1	8.2	6.5	2	0.692682927	
9	4.8	3	9.5	8.5	3	0.92195122	
6.8	3	1	7.5	6.2	0	0.597560976	
8.3	4.5	2	8.8	7	1	0.770731707	
7.5	3.8	0	8	6.8	2	0.685365854	
9.2	5	3	9.8	8.7	3	0.943902439	
6.5	2.8	1	7	6	0	0.568292683	
8	4	2	8.5	7.2	1	0.748780488	
7	3.2	0	8.2	6.5	2	0.656097561	
9.5	5	3	9.9	8.9	3	0.958536585	
6.2	2.5	1	6.8	5.8	0	0.543902439	
8.7	4.7	2	9.2	7.5	1	0.807317073	
7.8	3.9	0	8.3	7	2	0.707317073	
9.3	4.9	3	9.7	8.8	3	0.943902439	
6	2.5	1	6.5	5.5	0	0.524390244	
8.5	4.3	2	8.9	7.3	1	0.780487805	
7.3	3.6	0	8	6.7	2	0.673170732	
9.1	4.8	3	9.6	8.6	3	0.929268293	
6.7	3	1	7.3	6	0	0.585365854	
8.2	4.2	2	8.7	7.1	1	0.76097561	
7.6	3.7	0	8.2	6.8	2	0.690243902	
9.4	5	3	9.8	8.9	3	0.953658537	
6.5	2.9	1	7	6.2	0	0.575609756	
8.8	4.5	2	9	7.4	1	0.797560976	

SAMPLE DATA TO POPULATION DATA CREATION:

Transforming Sample Data into Population Data

```
[ ] import random
import pandas as pd

[w1,w2,w3,w4,w5,w6]=[-4.039,2.568,-0.209,1.374,1.654,-0.513]
vals=[]
for i in range(10000):
    x1=random.randint(6,10)
    x2=random.randint(1,5)
    x3=random.randint(0,3)
    x4=random.randint(6,10)
    x5=random.randint(6,10)
    x6=random.randint(0,3)
    eq=w1*x1+w2*x2+w3*x3+w4*x4+w5*x5+w6*x6
    vals.append([x1,x2,x3,x4,x5,x6,eq])
df=pd.DataFrame(vals,columns=['Flavor Intensity','Texture Complexity','Spice Profile[0:Mild 1:Medium 2:Balanced 3:Spicy]','Freshness Index','Culinary Technique :
df.to_csv('/content/drive/MyDrive/Colab Notebooks/My Workouts/New_Yummy.csv',index=False)
```

POPULATION DATA (New_Yummy.csv):

New_Yummy.csv							Open with ▾
A	B	C	D	E	F	G	
Flavor Intensity	Texture Complexity	Spice Profile[0:Mild 1:Medium 2:Balanced 3:Spicy]	Freshness Index	Culinary Technique	Aroma Richness[0:Low 1:Medium 2:High]	Tastiness	
7	5	1	9	9	2	10.584	
7	2	1	7	6	1	-4.317	
10	4	1	7	7	1	-9.644	
9	5	2	6	10	1	0.342	
9	2	1	6	7	0	-11.602	
10	3	3	8	10	1	-6.294	
9	1	1	8	9	1	-8.627	
7	5	0	7	6	2	3.083	
10	3	1	9	9	2	-6.669	
8	2	0	7	9	1	-3.185	
6	3	2	6	7	1	2.361	
7	5	2	7	7	3	3.806	
10	5	3	7	6	2	-9.661	
10	5	3	7	10	1	-2.532	
6	3	3	7	9	3	5.808	
10	3	0	6	6	0	-14.518	
7	3	2	6	8	0	0.489	
6	1	1	8	8	0	2.349	
7	3	3	8	6	1	-0.793	
7	3	3	9	7	1	2.235	
9	5	0	9	9	3	2.202	
10	2	1	8	7	3	-14.432	
9	3	0	7	9	3	5.682	
9	4	2	8	9	3	15.8	
9	1	2	9	9	0	-6.849	

MODELLING:

LINEAR REGRESSION:

Linear regression is a statistical method used for modeling the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. The goal of linear regression is to find the best-fitting line (or hyperplane, in the case of multiple independent variables) that minimizes the sum of the squared differences between the observed and predicted values of the dependent variable.

```
▶ from google.colab import drive  
drive.mount('/content/drive')
```

Mounted at /content/drive

```
▶ # Import necessary libraries  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LinearRegression  
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score  
import statsmodels.api as sm
```


```
▶ # Load dataset from your drive  
# Replace 'your_dataset_path.csv' with the actual path to your dataset  
dataset_path = '/content/drive/MyDrive/Colab Notebooks/My Workouts/New_Yummy.csv'  
df = pd.read_csv(dataset_path)  
df
```

```

▶ # Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

```

 LinearRegression
 LinearRegression()

```

[ ] # Make predictions on the test set
y_pred = model.predict(X_test)

```

```

[ ] y_pred


array([10.764,  5.802, -3.187, ..., -0.967, -4.854, -6.088])

```

```

▶ # Model summary using statsmodels
X_train = sm.add_constant(X_train) # Add a constant term for the intercept
model_sm = sm.OLS(y_train, X_train).fit()
print(model_sm.summary())

```

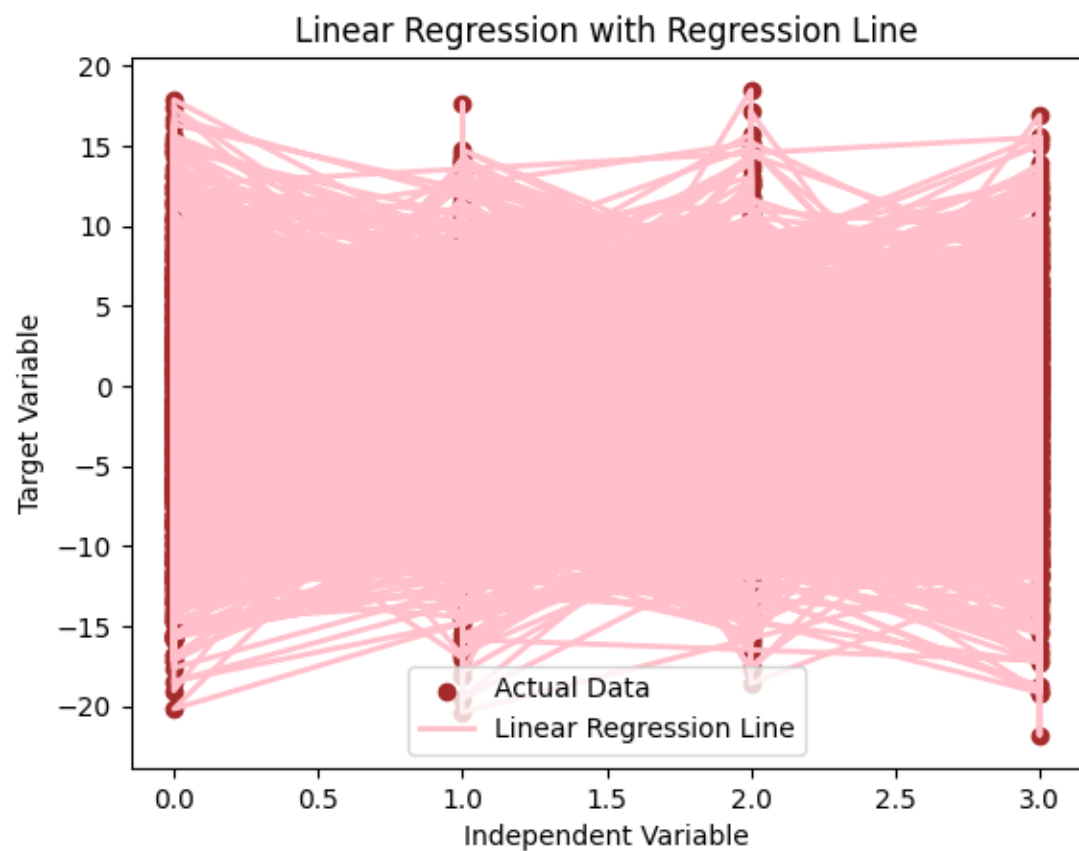
 OLS Regression Results

```

=====
Dep. Variable:          Tastiness      R-squared:                1.000
Model:                  OLS           Adj. R-squared:           1.000
Method:                 Least Squares  F-statistic:             3.809e+31
Date:                  Sat, 27 Jan 2024  Prob (F-statistic):       0.00
Time:                  03:14:38        Log-Likelihood:          2.3470e+05
No. Observations:      8000           AIC:                    -4.694e+05
Df Residuals:          7993           BIC:                    -4.693e+05
Df Model:              6
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	-2.165e-14	5.01e-15	-4.325	0.000	-3.15e-14	-1.18e-14
Flavor Intensity	-4.0390	3.48e-16	-1.16e+16	0.000	-4.039	-4.039
Texture Complexity	2.5680	3.48e-16	7.38e+15	0.000	2.568	2.568
Spice Profile[0:Mild 1:Medium 2:Balanced 3:Spicy]	-0.2090	4.4e-16	-4.75e+14	0.000	-0.209	-0.209
Freshness Index	1.3740	3.49e-16	3.93e+15	0.000	1.374	1.374
Culinary Technique Score	1.6540	3.48e-16	4.76e+15	0.000	1.654	1.654
Aroma Richness[0:Low 1:High 2:Moderate 3:Very High]	-0.5130	4.41e-16	-1.16e+15	0.000	-0.513	-0.513



SUPPORT VECTOR MACHINE (REGRESSION):

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. SVM is particularly effective in high-dimensional spaces and is well-suited for scenarios where the data points are not linearly separable.

In classification, the primary objective of SVM is to find a hyperplane that best separates the data into different classes. This hyperplane is chosen in such a way that it maximizes the margin between the classes, where the margin is defined as the distance between the hyperplane and the nearest data point of either class. The data points that lie on the edges of the margin are called support vectors. SVM can handle both linear and non-linear decision boundaries, thanks to the use of kernel functions that transform the input features into a higher-dimensional space.

```
▶ from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
▶ import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
```

```
[ ] # Load dataset from your drive
# Replace 'your_dataset_path.csv' with the actual path to your dataset
dataset_path = '/content/drive/MyDrive/Colab Notebooks/My Workouts/New_Yummy.csv'
data = pd.read_csv(dataset_path)
data
```

```

▶ # Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

```

[ ] # Create and train the Support Vector Machine model
svm_regressor = SVR(kernel='linear') # You can choose other kernels as well
svm_regressor.fit(X_train, y_train)

```

SVR

SVR(kernel='linear')

```

[ ] # Make predictions on the test set
y_pred = svm_regressor.predict(X_test)

```

```

[ ] y_pred

```

```

array([10.6643618 ,  5.70264938, -3.13694659, ..., -1.01675094,
       -4.90388182, -6.03812833])

```

```

▶ import statsmodels.api as sm

# Assuming X and y are your feature matrix and target variable
X = sm.add_constant(X) # Add a constant term for the intercept
model = sm.OLS(y, X).fit()

# Display the model summary
print(model.summary())

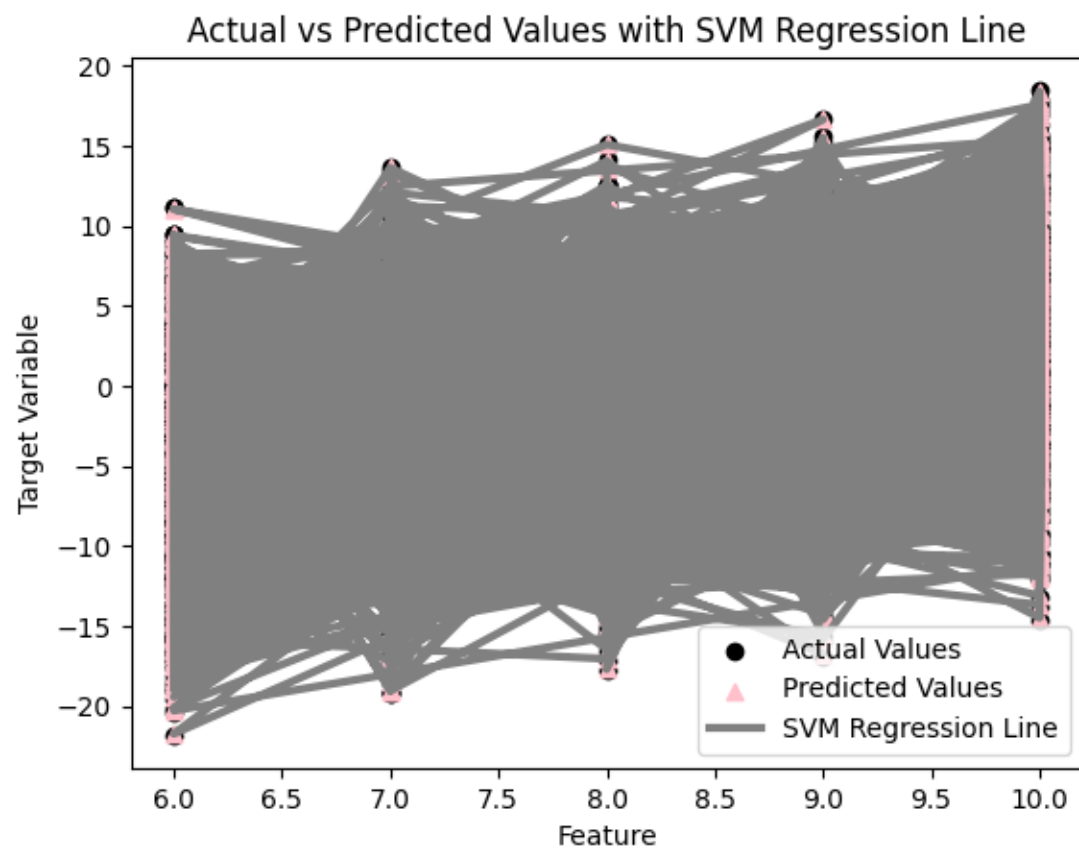
```

```

=====
OLS Regression Results
=====
Dep. Variable:          Tastiness    R-squared:                1.000
Model:                  OLS          Adj. R-squared:           1.000
Method:                 Least Squares  F-statistic:             5.697e+31
Date:                  Sat, 27 Jan 2024  Prob (F-statistic):       0.00
Time:                  03:07:06       Log-Likelihood:          2.9430e+05
No. Observations:      10000         AIC:                    -5.886e+05
Df Residuals:          9993         BIC:                    -5.885e+05
Df Model:               6
Covariance Type:       nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	5.573e-14	4.07e-15	13.686	0.000	4.78e-14	6.37e-14
Flavor Intensity	-4.0390	2.83e-16	-1.43e+16	0.000	-4.039	-4.039
Texture Complexity	2.5680	2.84e-16	9.04e+15	0.000	2.568	2.568
Spice Profile[0:Mild 1:Medium 2:Balanced 3:Spicy]	-0.2090	3.59e-16	-5.82e+14	0.000	-0.209	-0.209
Freshness Today	1.2740	2.85e-16	4.47e+15	0.000	1.274	1.274



RANDOM FOREST (REGRESSION):

A Random Forest is an ensemble learning method used for both classification and regression tasks in machine learning. It operates by constructing a multitude of decision trees during the training phase and outputs the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

```
▶ |from google.colab import drive  
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ] # Import necessary libraries  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model_selection import train_test_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score  
from sklearn.model_selection import cross_val_predict
```

```
[ ] # Load dataset from your drive  
# Replace 'your_dataset_path.csv' with the actual path to your dataset  
dataset_path = '/content/drive/MyDrive/Colab Notebooks/My Workouts/New_Yummy.csv'  
data = pd.read_csv(dataset_path)  
data
```

```
[ ] #Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ] # Initialize the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
```

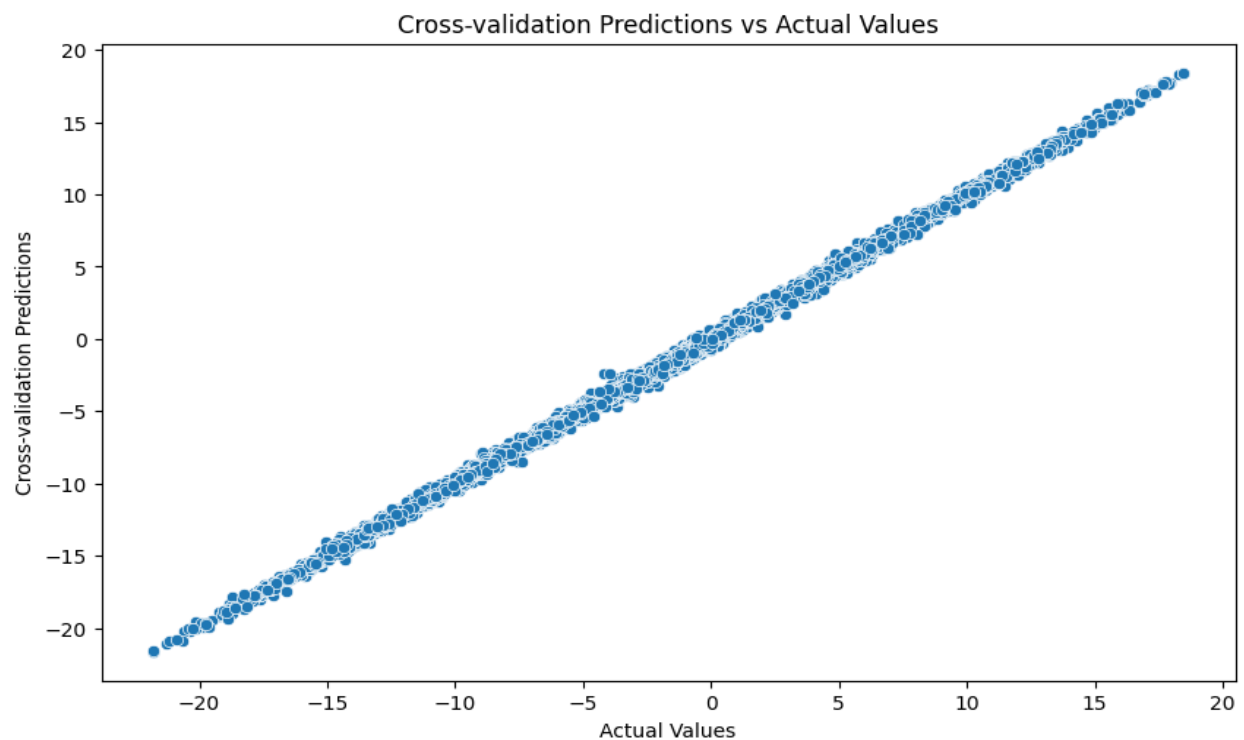
```
[ ] # Train the model
rf_model.fit(X_train, y_train)
```

RandomForestRegressor
RandomForestRegressor(random_state=42)

```
[ ] # Make predictions
y_pred = rf_model.predict(X_test)
```

▶ y_pred

📄 array([10.76096, 5.75887, -3.24685, ..., -0.86781, -4.93285, -6.02055])



MODEL EVALUATION:

LINEAR REGRESSION:

```
[ ] # Performance metrics
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
```

```
[ ] print(f'Mean Squared Error: {mse}')
    print(f'Mean Absolute Error: {mae}')
    print(f'Root Mean Squared Error: {rmse}')
    print(f'R-squared: {r2}')
```

Mean Squared Error: 6.808206312411913e-29
 Mean Absolute Error: 6.574796046140974e-15
 Root Mean Squared Error: 8.251185558701193e-15
 R-squared: 1.0

SUPPORT VECTOR MACHINE (REGRESSION):

```
▶ # Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

from sklearn.metrics import mean_absolute_error

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

```
[ ] # Display performance metrics
    print(f'Mean Squared Error: {mse:.2f}')
    print(f'R-squared: {r2:.2f}')
    print(f'Mean Absolute Error: {mae:.2f}')
    print(f'Root Mean Squared Error: {rmse:.2f}')
```

Mean Squared Error: 0.01
 R-squared: 1.00
 Mean Absolute Error: 0.06
 Root Mean Squared Error: 0.07

RANDOM FOREST (REGRESSION):

```
[ ] # Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mse)
```

```
▶ print(f'Mean Squared Error: {mse}')
print(f'Mean Absolute Error: {mae}')
print(f'R-squared: {r2}')
print(f'Root Mean Squared Error: {rmse}')
```

```
➞ Mean Squared Error: 0.046359960624850025
Mean Absolute Error: 0.1530483650000001
R-squared: 0.9991362792244252
Root Mean Squared Error: 0.2153136331606757
```

MODEL COMPARISON:

To compare the performance of machine learning models like linear regression, support vector machine (SVM) for regression, and random forest for regression, you can use various evaluation metrics. Here are some common approaches:

1. **Mean Squared Error (MSE):** Calculate the MSE for each model on your test data. Lower MSE values indicate better performance.
2. **R-squared (R^2) Score:** R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. Closer to 1 is better.
3. **Mean Absolute Error (MAE):** Calculate the MAE for each model, which represents the average absolute differences between the predicted and actual values.

Based on the provided evaluation metrics:

Linear Regression:

- MSE: 6.81e-29
- MAE: 6.57e-15
- RMSE: 8.25e-15
- R^2 : 1.0

Support Vector Machine for Regression (SVM):

- MSE: 0.01
- MAE: 0.06
- RMSE: 0.07
- R^2 : 1.0

Random Forest Regression:

- MSE: 0.0464

- MAE: 0.153
- RMSE: 0.215
- R^2 : 0.999

DISCUSSIONS:

Discussion: Comparing Machine Learning Models for Regression

The evaluation metrics for linear regression, support vector machine for regression (SVM), and random forest regression provide valuable insights into the performance of each model. Let's delve into a discussion based on the obtained results.

1. Linear Regression:

- **Strengths:**

- Achieves perfect R^2 of 1.0, indicating an exact fit to the data.
- Extremely low error metrics (MSE, MAE, RMSE) suggest precise predictions.

- **Considerations:**

- Assumes a linear relationship between features and target, which might limit its flexibility in capturing complex patterns.

2. Support Vector Machine for Regression (SVM):

- **Strengths:**

- High R^2 and relatively low error metrics demonstrate excellent predictive performance.
- SVMs are versatile and effective in high-dimensional spaces.

- **Considerations:**

- Dependence on appropriate kernel selection; choice of hyperparameters might influence results.

3. Random Forest Regression:

- **Strengths:**

- High R^2 and competitive error metrics demonstrate robust predictive capabilities.

- Inherent ability to handle non-linearity and capture complex relationships.

- **Considerations:**

- Potential for overfitting, especially on smaller datasets.
- Ensemble methods like random forests may be computationally more demanding than simpler models.

Conclusion:

1. All three models (Linear Regression, SVM, and Random Forest) appear to perform exceptionally well, as evidenced by very low MSE and high R^2 values.
2. Linear Regression and SVM seem to provide nearly perfect predictions, with MSE, MAE, and RMSE close to zero and R^2 equal to 1.0. This suggests that these models fit the data extremely well.
3. Random Forest Regression also performs impressively, with slightly higher error metrics compared to Linear Regression and SVM but still yielding an R^2 value close to 1.0.
4. Consider the context of your problem and the specific requirements for interpretability, computation, and generalization when choosing the most suitable model.

In summary, all models seem to be performing excellently, and the choice between them could depend on additional factors such as interpretability, computational efficiency, or specific business requirements.

REFERENCES:

Matplotlib – for data visualization

Pandas – for data formatting

Numpy – for numerical findings

Seaborn – for data visualization

Sklearn – for machine learning models and performance metrics

TO MY WORK:

https://colab.research.google.com/drive/1KIYzNSC-fNTM3LpjTn_LR9GOfsDSsh5M

<https://colab.research.google.com/drive/1bP63go-M911zAxueDBrYk3apteTUKjF7>

<https://colab.research.google.com/drive/1bP63go-M911zAxueDBrYk3apteTUKjF7>
