



**בית ספר:** בسمת החדש

**שם תלמיד:** יאנה קורוטקוב

**תעודת זהות:** 345742241

**תאריך:** 12.05.2025

**שנת לימודים:** תשפ"ה

**שם המורה:** דודקין סבטלנה

<https://colab.research.google.com/drive/13UVZLGx03N5HhkjvdfLpfJJfdhQFaiRc?usp=sharing>



**Banana Quality**

## מטרת המחבר

מטרת המחבר היא לפתח ולבוחן מודלים של למידת מכונה לניבוי הבשלות של בננות בתבסס על מאפיינים כמו גודל, משקל, בשלות, מתיקות, רכות, זמן קטיף וחומציות. לשם כך, נבחרו שני אלגוריתמים עיקריים: רגרסיה לוגיסטיבית-KNN. המחבר שואף להשוות בין המודלים במנוחים של דיקט, שליפה ויעילות, ובכך לזרה את האלגוריתם המיטב למשימה זו. התוצאות יכולות לשיפור תהליכי מיון ואחסון בתעשיית הפירות.

## תיאור הנתונים

מאגר הנתונים מכיל מידע על 8000 בננות, אשר מסוגות לפי מאפיינים שונים כגון מגוון, רמת בשלות ופרמטרים נוספים. הנתונים כוללים תכונות שונות שניתן להשתמש בהן לניתוח וויזי.

מאגר הנתונים הורד מפלטפורמת Kaggle והוא מיועד לאימון מודלים של למידת מכונה המתמקד במשימות סיווג וחיזוי.

**קישור למאגר הנתונים:**

<https://www.kaggle.com/datasets/l3llff/banana?resource=download>

**פירוט על סוג הנתונים במאגר:**

במאגר הנתונים ישנו 8000 שורות ו-8 עמודות

- **Size** - גודל בנהנה
- **Weight** - משקל בנהנה
- **Sweetness** - מתיקות בנהנה
- **Softness** - רכות בנהנה
- **HarvestTime** - זמן שעבר מאז הקטיף
- **Ripeness** - בשלות בנהנה
- **Acidity** - חומציות בנהנה
- **Quality** - איכות בנהנה

## חקר נתונים

טיענות נתונים ותצוגה של הנתונים מהמגר:

```
df = pd.read_csv('banana_quality.csv')
df.head()
```

	Size	Weight	Sweetness	Softness	HarvestTime	Ripeness	Acidity	Quality
0	-1.924968	0.468078	3.077832	-1.472177	0.294799	2.435570	0.271290	Good
1	-2.409751	0.486870	0.346921	-2.495099	-0.892213	2.067549	0.307325	Good
2	-0.357607	1.483176	1.568452	-2.645145	-0.647267	3.090643	1.427322	Good
3	-0.868524	1.566201	1.889605	-1.273761	-1.006278	1.873001	0.477862	Good
4	0.651825	1.319199	-0.022459	-1.209709	-1.430692	1.078345	2.812442	Good

:DataFrame ב – DataFrame

```
[ ] missing_values = df.isnull()
missing_counts = missing_values.sum()
print(missing_counts)
```

	Size	Weight	Sweetness	Softness	HarvestTime	Ripeness	Acidity	Quality
	0	0	0	0	0	0	0	0

ניתן לראות שלא קיימים ערכים חסרים במאגר.

הציג נתונים בסיסי נתונים: כמה סוגי (קטגוריות), מספר הרשומות השלמות בכל עמודה, כמה רשומות של כל סיווג, נתונים חסרים

df.unique()		df.info()			
<code>0</code>		<class 'pandas.core.frame.DataFrame'> RangeIndex: 8000 entries, 0 to 7999 Data columns (total 8 columns):			
<code>Size</code> 8000		# Column Non-Null Count Dtype			
<code>Weight</code> 8000		---			
<code>Sweetness</code> 8000		0 Size 8000 non-null float64			
<code>Softness</code> 8000		1 Weight 8000 non-null float64			
<code>HarvestTime</code> 8000		2 Sweetness 8000 non-null float64			
<code>Ripeness</code> 8000		3 Softness 8000 non-null float64			
<code>Acidity</code> 8000		4 HarvestTime 8000 non-null float64			
<code>Quality</code> 2		5 Ripeness 8000 non-null float64			
		6 Acidity 8000 non-null float64			
		7 Quality 8000 non-null object			
		dtypes: float64(7), object(1)			
		memory usage: 500.1+ KB			

בדיקה כמה ערכים שונים יש בכל עמודה:

תיאור הנתונים: ממוצעים, סטיות תקן, מינימום, חיצוניים, מקסימום:

df.describe()							
	Size	Weight	Sweetness	Softness	HarvestTime	Ripeness	Acidity
count	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000
mean	-0.747802	-0.761019	-0.770224	-0.014441	-0.751288	0.781098	0.008725
std	2.136023	2.015934	1.948455	2.065216	1.996661	2.114289	2.293467
min	-7.998074	-8.283002	-6.434022	-6.959320	-7.570008	-7.423155	-8.226977
25%	-2.277651	-2.223574	-2.107329	-1.590458	-2.120659	-0.574226	-1.629450
50%	-0.897514	-0.868659	-1.020673	0.202644	-0.934192	0.964952	0.098735
75%	0.654216	0.775491	0.311048	1.547120	0.507326	2.261650	1.682063
max	7.970800	5.679692	7.539374	8.241555	6.293280	7.249034	7.411633

המרת את הערכים בעמודה "איכות" מקטגוריאליים לערכים מספריים ('טוב' – 1, 'רע' – 0) כדי להקל על עבודה המודל

```
df['Quality'] = df['Quality'].map({'Good': 1, 'Bad': 0})  
df
```

## GroupBy

```
df.groupby('Quality')['HarvestTime'].mean()
```

HarvestTime

Quality

0	-1.504313
1	-0.000519

dtype: float64

חשבתי את זמן הקטיף הממוצע לכל קבוצת איכות. במבנה באיכות נמוכה זמן הקטיף היה נמוך משמעותית מהממוצע, בעוד שבבננות באיכות גבוהה הוא היה קרוב לממוצע. תוצאה זו מחזקת את הקשר בין קטיף מוקדם לבין ירידה באיכות הבננות

```
df.groupby('Quality')['Ripeness'].agg(['mean', 'median'])
```

mean median

Quality

0	0.038967	0.069118
1	1.521007	1.515116

חשבתי את רמת הבשלות הממוצעת לכל קבוצת איכות. במבנה באיכות נמוכה רמת הבשלות הממוצעת הייתה כמעט אפס, בעוד שבבננות באיכות גבוהה היא הייתה בסביבות 1.5. תוצאה זו מחזקת את הקשר בין רמת בשלות גבוהה לבין איכות פרי טובה

## ויזואליזציה של הנתונים

ספריות הנדרשות להרצתה:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.utils import shuffle
```

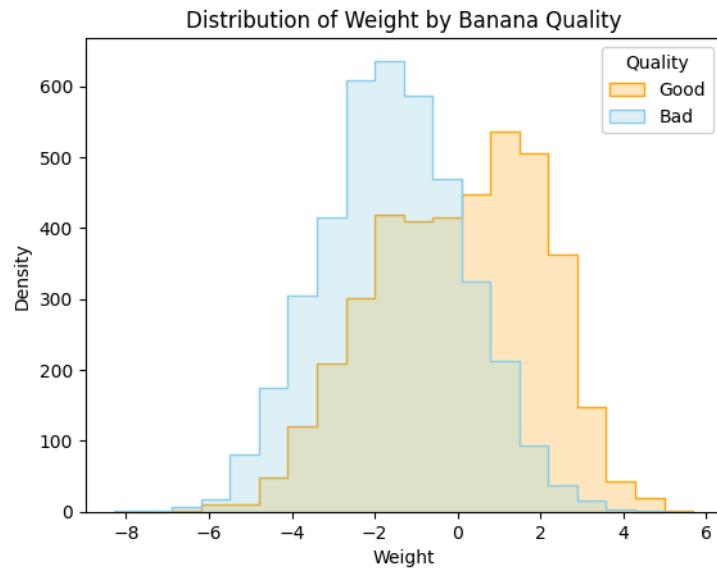
	Size	Weight	Sweetness	Softness	HarvestTime	Ripeness	Acidity	Quality
0	-1.924968	0.468078	3.077832	-1.472177	0.294799	2.435570	0.271290	1
1	-2.409751	0.486870	0.346921	-2.495099	-0.892213	2.067549	0.307325	1
2	-0.357607	1.483176	1.568452	-2.645145	-0.647267	3.090643	1.427322	1
3	-0.868524	1.566201	1.889605	-1.273761	-1.006278	1.873001	0.477862	1
4	0.651825	1.319199	-0.022459	-1.209709	-1.430692	1.078345	2.812442	1
...	...	...	...	...	...	...	...	...
7995	-6.414403	0.723565	1.134953	2.952763	0.297928	-0.156946	2.398091	0
7996	0.851143	-2.217875	-2.812175	0.489249	-1.323410	-2.316883	2.113136	0
7997	1.422722	-1.907665	-2.532364	0.964976	-0.562375	-1.834765	0.697361	0
7998	-2.131904	-2.742600	-1.008029	2.126946	-0.802632	-3.580266	0.423569	0
7999	-2.660879	-2.044666	0.159026	1.499706	-1.581856	-1.605859	1.435644	0

8000 rows × 8 columns

המאפיינים עברו סטנדרטיזציה כדי להכין אותם למודלים. בתהליך זה, הממוצע של כל عمودה הושב ל-0 והסטיית תקן ל-1. ערכים חיוביים מייצגים נתונים מעל הממוצע, וערכים שליליים מתחתן. כך כל מאפיין מקבל חשיבות שווה במודל

## Distribution of Banana Weight by Quality

```
sns.histplot(data = df, x = 'Weight', hue = 'Quality', element = 'step', bins = 20, palette = {0: 'skyblue', 1: 'orange'})  
  
plt.title("Distribution of Weight by Banana Quality")  
plt.xlabel("Weight")  
plt.ylabel("Density")  
plt.legend(title = 'Quality', labels = ['Good', 'Bad'])  
plt.show()
```



הגרף מציג את ההתפלגות של משקל הבננות לפי איכות

**צירוי הגרף:**

- X – משקל הבננה
- Y – כמות (צפיפות)
- כתום - בננות טובות, כחול - בננות רעות

**מה אפשר לראות בגרף?**

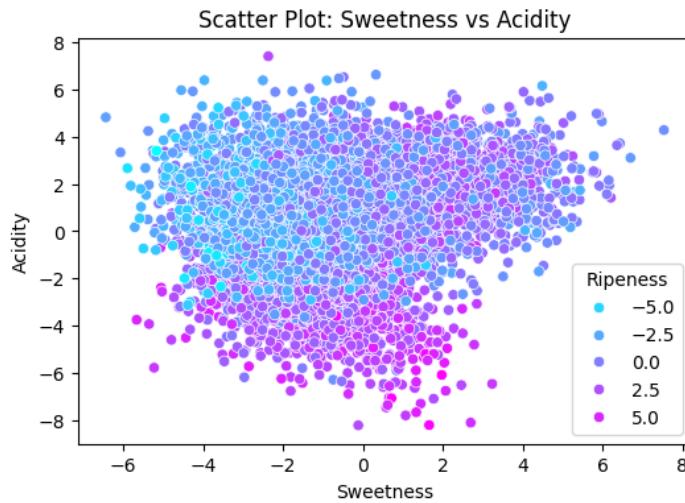
- הקו הכתום מרוכז יותר מצד ימין, כלומר לבננות טובות יש משקל גבוה יותר
- הקו הכחול מרוכז יותר מצד שמאל, כלומר לבננות רעות נוטות לשקל פחות
- יש קצת חפיפה בין הקבוצות, אבל אפשר לראות הבדל ברור בהתפלגות

**מסקנה**

בננות באיכות גבוהה שוקלות יותר ממוצע מבनנות באיכות נמוכה. לכן, המשקל הוא מאפיין שיכול לעזור לנביא את איכות הבננה

## Scatter Plot of Sweetness vs. Acidity Colored by Ripeness

```
plt.figure(figsize=(6, 4))
sns.scatterplot(x='Sweetness', y='Acidity', data=df, hue='Ripeness', palette='cool')
plt.title('Scatter Plot: Sweetness vs Acidity')
plt.xlabel('Sweetness')
plt.ylabel('Acidity')
plt.show()
```



הграф מציג את הקשר בין מתיקות לבין חומציות של בננות

מה הנקודות מייצגות?

- כל נקודה מייצגת בננה אחת
- הצבע של הנקודה מראה את דרגת הבשלות: נקודות כחולות - בננות פחות בשלות. נקודות סגולות וורודות - בננות בשנות יותר

מה אפשר לראות בגרף?

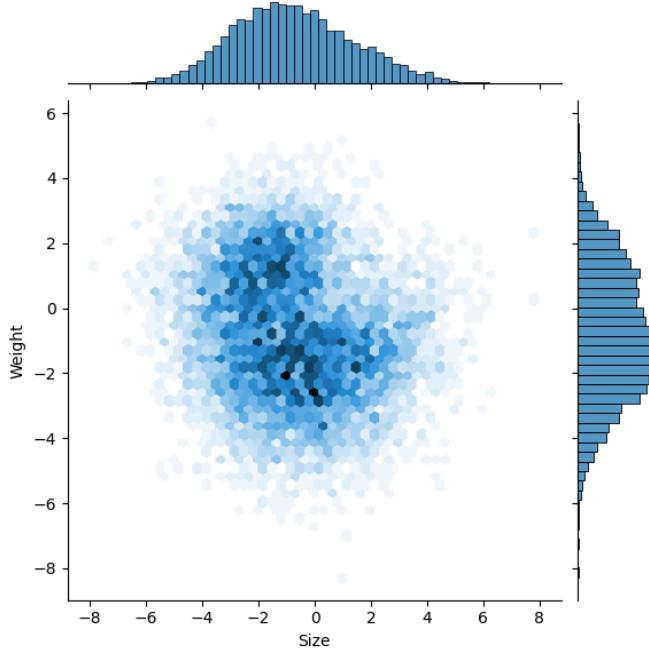
- הנקודות הכחולות (בננות פחות בשלות) נמצאות לרוב בחלק העליון של הגרף, שבו החומציות גבוהות יותר. לעומת זאת, בננות פחות בשנות יותר חמימות במעט, שבו המתיקות נמוכה יותר. זה אומר שבנן פחות בשנות הן פחות מתוקות
- ככל שהבננה מבשילה, היא נעשית פחות חמוצה יותר מתוקה, וכך הנקודות הסגולות והורודות צוזות כלפי מטה ימינה

מסקנה

בנן פחות בשנות (נקודות כחולות) הן יותר חמימות ופחות מתוקות, ואילו עם הבשלתן הן הופכות פחות חמימות וייתר מתוקות. זה הגיוני, כי במהלך ההבשלת העמילן הופך לסוכר, והחומציות יורדת

## Hexbin Plot of Size vs. Weight

```
sns.jointplot(x=df['Size'], y=df['Weight'], kind="hex")
plt.show()
```



הגרף זהה מציג את הקשר בין גודל הבננה לבין משקל שלה

**ציר הגרפי:**

- **X** – גודל הבננה
- **Y** – משקל הבננה
- הצביע והצפיפות של הנקודות מראים כמה בננות יש בכל אזור. ככל שהצביע כהה יותר, יש יותר בננות עם אותם נתונים

**מה ניתן לראות בגרף?**

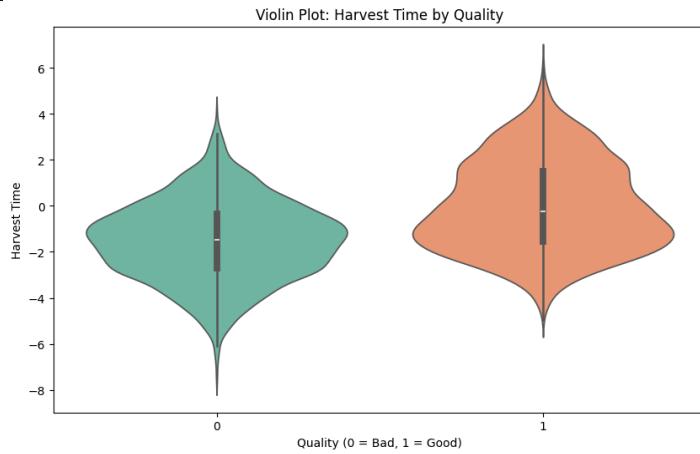
- רוב הבננות מרכזיות במרכז, קרוב לאפס, מה שאומר שהגדלים והמשקלות שלهن קרובים למוצע
- יש אזורים עם צבע כהה יותר – אלה הערכים הנפוצים ביותר
- ככל שמתרחקים מהמרכז, יש פחות בננות עם הערכים האלה
- בצד הימני מופיעות היסטוגרמות, שמראות אילו גדלים ומשקלים מופיעים הכי הרבה

**מסקנה**

יש קשר בין גודל הבננה למשקל שלה: ככל שהבננה גדולה יותר, כך לרוב היא גם כבדה יותר. אבל יש גם יוצאים מן הכלל, ולא כל הבננות באותו גודל שוקלות אותו דבר

## Violin Plot of Harvest Time by Quality

```
plt.figure(figsize=(10, 6))
sns.violinplot(x='Quality', y='HarvestTime', data=df, palette='Set2')
plt.title('Violin Plot: Harvest Time by Quality')
plt.xlabel('Quality (0 = Bad, 1 = Good)')
plt.ylabel('Harvest Time')
plt.show()
```



הגרף מציג את ההתפלגות של זמן הקטיף בקרב בנןות באיכות טוביה לעומת בנןות באיכות גרועה

המלבן האפור במרכז הגרף מייצג את החציון (הקו האופקי) ואת הטווח שבין הרביעון הראשון לרביעון השלישי. כמו כן, הוא מסמן את האזור שבו נמצאים 50% מהנתונים המרכזים של כל קבוצה

**ציר הגרף:**

- X – צפיפות הנתונים
- Y – זמן הקטיף

**מה ניתן לראות בגרף?**

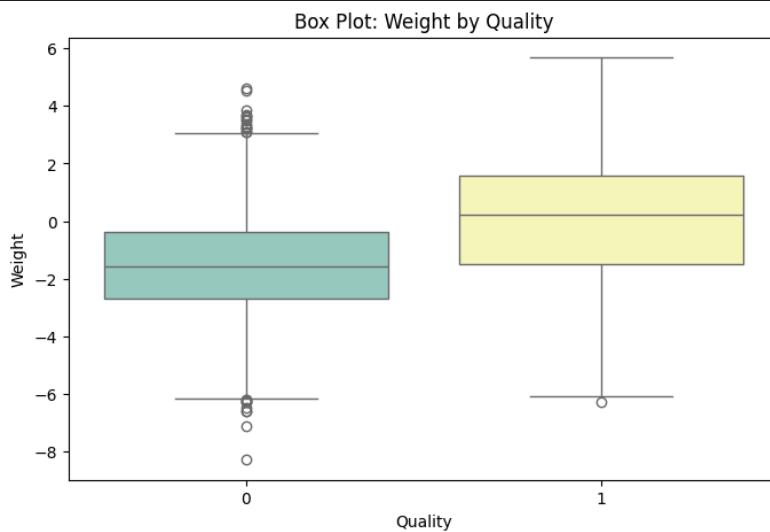
- בbananaות באיכות גבוהה, זמן הקטיף מתרცץ בעיקר סביב ערכים מסוימים, בעוד שבbananaות באיכות נמוכה הוא מפוזר על פני טווח רחב יותר
- בbananaות האיכותיות יש שני ריכוזים עיקריים של זמן קטיף, מה שעשוי לרמז על שתי תקופות אידיאליות לקטיף או על הבדל בתנאי הגידול המשפיעים על איכות הבananות

**מסקנה**

יש קשר בין זמן הקטיף לאיכות הבananות:bananaות שנקטפו בזמן הבשלה אידיאלי נוטות להיות באיכות טוביה יותר, בעוד שקטיף מוקדם או בזמן מאוחר עלול להוביל לאיכות ירודה

## Box Plot of Weight by Quality

```
plt.figure(figsize=(8, 5))
sns.boxplot(x="Quality", y="Weight", data=df, palette="Set3")
plt.title("Box Plot: Weight by Quality")
plt.xlabel("Quality")
plt.ylabel("Weight")
plt.show()
```



הגרף מציג את הקשר בין משקל הבננה לבין איכות הבננה. המלבן מציג את טווח המשקל בין הרביעון השני לשישי (50% מהבננות). הקווים שבתוך המלבן הוא החציון. הקווים שמחוץ למלבן ("זקינים") מראים את המשקל הנמוך והגבוה שנחשב רגיל. נקודות מחוץ לטווח זהה הן חריגות – בננות קלות או כבדות במיוחד

**צירי הגרפי:**

- **X** – איכות (0 = בננות רעות, 1 = בננות טובות)
- **Y** – משקל

**מה ניתן לראות בגרף?**

- לבנות טובות יש משקל גבוה יותר בדרך כלל
- אצל בננות רעות יש פיזור גדול יותר, כולל הרבה בננות קלות מאוד

**מסקנה**

משקל הבננה קשור לאיכות שלה – בננות טובות נוטות להיות כבדות יותר ויציבות יותר מאשר בננות רעות.

## Correlation of Features with Quality

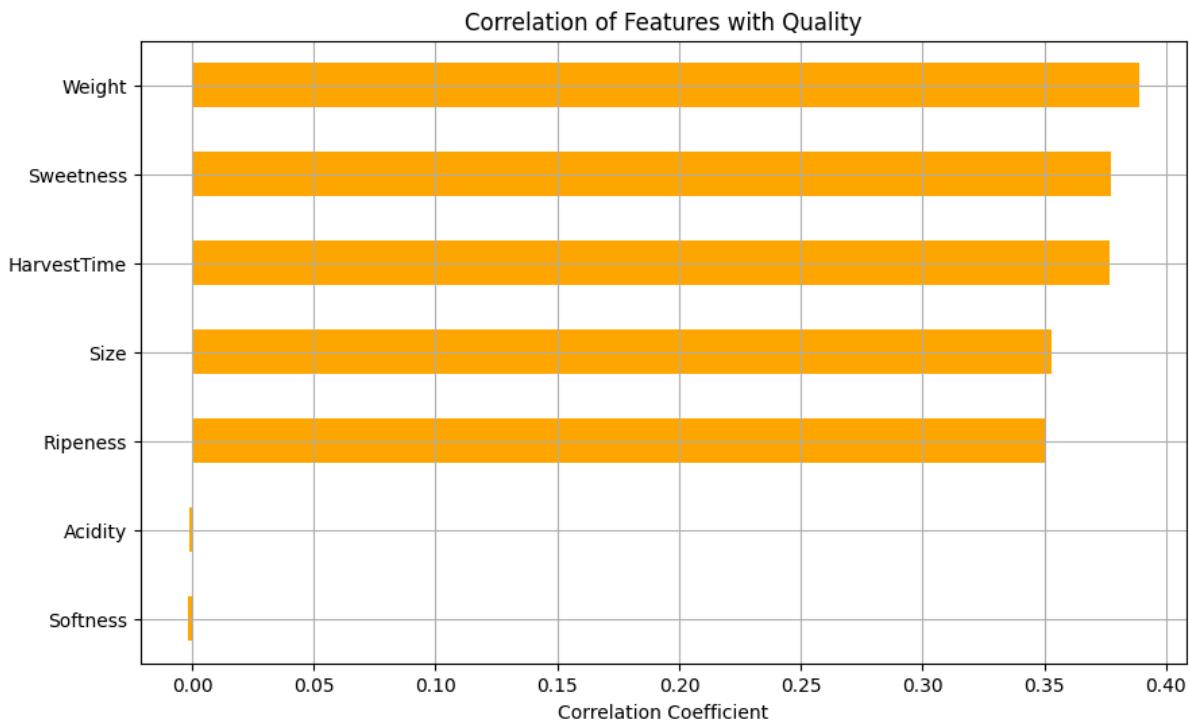
```
corr_matrix = df.corr()

target_corr = corr_matrix['Quality'].drop('Quality')

correlation_table = pd.DataFrame({
    'Feature': target_corr.index,
    'Correlation with Quality': target_corr.values
})

print(correlation_table)

plt.figure(figsize=(10, 6))
target_corr.sort_values().plot(kind='barh', color='orange')
plt.title('Correlation of Features with Quality')
plt.xlabel('Correlation Coefficient')
plt.grid(True)
plt.show()
```



אפשר לראות שרוב המאפיינים של הבננות מציגים מזגעים מתאימים חזק עם אינטראקציות הבננות, כלומר הם משפיעים על סיווג הבננות כ"טובות" או "גרועות". עם זאת, למאפיינים כמו רכות וחומציות יש מתאימים מאוד חלש, מה שמעיד שהם כמעט ולא משפיעים על אינטראקציות הבננות.

## למידת מכונה

המודלים אשר בחרתי להשתמש בהם בפרויקט

Logistic Regression

(K-Nearest Neighbors) KNN

### הכנות הנתונים

בחרתי את המאפיינים שיכולים להשפיע על תחזית איקотה של הבננה. במקורה של, מדובר בפרמטרים כמו גודל, משקל, מתיקות, בשלות, רכות וזמן קצר. נתונים אלו המרדי למספרים כדי שייהי אפשר להשתמש בהם באלאוריתמים של למידת מכונה. לדוגמה, משתנה קטגוריאלי איקות הבננה (טוב או רע) הוסב לערכים בינאריים: 1 עבור בנות טובות ו-0 עבור בנות רעות כל הנתונים כבר היו מותוקנים (סטנדרטיזציה), ולא היו ערכים חסרים, כך שלא היה צורך בטיפול נוסף

חלוקת הנתונים לקבוצות אימון, אימון ובדיקה

### הנתונים חולקו לשלושה חלקים

Train • שבעים אחוז מהנתונים לאימון המודלים

Test • שלושים אחוז מהנתונים להערכת הביצועים של המודל הסופי

```
df = shuffle(df, random_state = 39)

X = df.drop(['Quality'], axis = 1)
y = df['Quality']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=39)

print(f"Train set: {X_train.shape}")
print(f"Test set: {X_test.shape}")

Train set: (5600, 7)
Test set: (2400, 7)
```

בהתחלת חילוקת הנתונים לשלושה חלקים: אימון, ולידציה ובדיקה. במהלך העבודה החלטתי לוותר על קבוצת הולידציה, כיוון שהיא לא שימושה באף שלב בנייתה. כך יכולתי להוסיף יותר נתונים לקבוצת האימון ולשפר את דיקוק המודלים. קבוצת הבדיקה נשארה לצד והשתמשתי בה רק בשלב הסופי, כדי לבדוק את ביצועי המודל על נתונים שלא ראה קודם – מה שמאפשר לעירק את איקוטו בצורה אמינה

## Logistic Regression

המודל שבחरתי מבוסס על רגסיה לוגיסטי, אשר משמשת לפתור בעיות סיווג על ידי חישוב ההסתברות שהמשתנה המטרה שייך לאחת הקטגוריות. מודל זה עדיף על פני רגסיה לנארית בהקשר זה, כיוון שהוא מיועד במיוחד למשימות של סיווג ביןארי, בעוד שרגסיה לנארית מתאימה יותר למשימות של חיזוי ערכים רציפים

על מנת לשפר את ביצועי המודל, נעשה שימוש ב-`GridSearchCV` מסדריית `sklearn` לأتيור היפר-פרמטרים האופטימליים. שיטה זו מאפשרת לבצע חיפוש יסודי על פני מרחב פרמטרים ולבחר את השילוב שמניב את התוצאות הטובות ביותר על סט הנתונים המקוריים

לאחר אימון המודל, קיבלתי תוצאות עם דיקן של **0.8817**

קודם כל ביצעת חיפוש אוטומטי לפרמטרים הći טובים עבור מודל הלוגיסטי

```
parameters = {
    'C' : [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty' : ['l2']
}

gscv1 = GridSearchCV(LogisticRegression(), parameters, cv = 5)
gscv1.fit(X_train, y_train)
print(gscv1.best_params_)

{'C': 0.001, 'penalty': 'l2'}
```

אחר כך אימנתי את המודל עם הפרמטרים האלה ובדקתי את רמת הדיקן שלו

```
best_params = gscv1.best_params_

lr = LogisticRegression(C=best_params['C'], penalty=best_params['penalty'])
lr.fit(X_train, y_train)

score = lr.score(X_test, y_test)
print(f"{score:.4f}")

0.8817
```

לאחר מכון בניתי ואימנתי מודל עם הפרמטרים אשר מצאתי

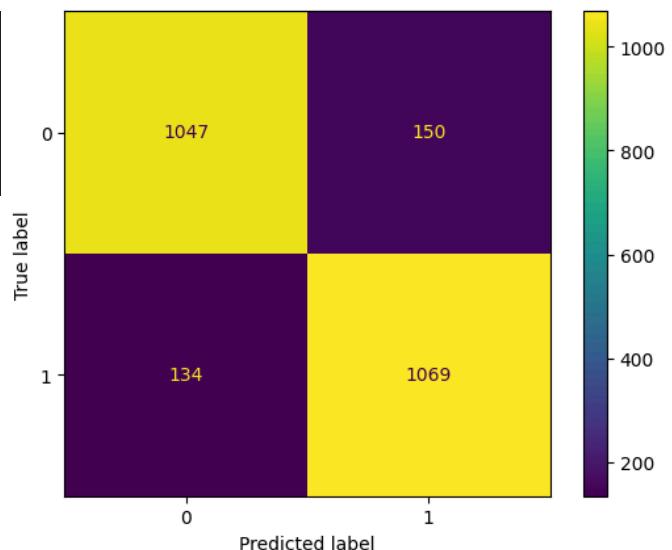
```
cm_lr = confusion_matrix(y_test, lr.predict(X_test))
print(cm_lr)

[[1047 150]
 [ 134 1069]]
```

## הציגי מטריצת בלבול כדי לראות אם המודל טועה

```
disp = ConfusionMatrixDisplay(cm_lr)
disp.plot()

plt.show()
```



מטריצת בלבול מציגה בקורסיה את הביצועים של המודל, על ידי השוואת בין התוצאות האמיטיות לתחזיות של המודל  
היא מחולקת לארבעה חלקים:

- המודל זיהה נכון בנבנה טוב – **True Positive**
- המודל זיהה נכון בנבנה גרועה – **True Negative**
- המודל חשב בטעות שבננה גרועה היא טוב – **False Positive**
- המודל חשב בטעות שבננה טוב היא גרועה – **False Negative**

ולבסוף חישבתי את כל מדדי ההערכתה של המודל:

```
pred1 = lr.predict(X_test)
print(classification_report(y_test, pred1, digits=4))

precision    recall  f1-score   support

      0       0.8865    0.8747    0.8806     1197
      1       0.8769    0.8886    0.8827     1203

  accuracy                           0.8817    2400
    macro avg       0.8817    0.8816    0.8817    2400
weighted avg       0.8817    0.8817    0.8817    2400
```

: אחוז החיזויים החיוביים הנכונים מתוך כל החיזויים החיוביים **Precision**

: אחוז החיזויים החיוביים הנכונים מתוך כל הדגימות החיוביות האמיטיות **Recall**

: ממוצע הרמוני בין דיקון ושליפה **F1-score**

: מספר הדגימות עבור כל קטגוריה **Support**

## KNN

המודל הזה פועל בכך שהוא בודק את סיווגם של K השכנים הקרובים ביותר לנಕודה חדשה ומחליט את סיווגה בהתאם לעלייהם. בהתחלה, חישבתי את הערך האופטימלי של K על מנת לשפר את דיוק המודל, באמצעות קוד שהרייצ' את הערכים השונים של K ובודק את דיוק המודל לכל ערך

לאחר אימון המודל, קיבלתי תוצאות עם דיוק של **0.9796**

בעזרת GridSearchCV בחרתי את הפרמטרים הći טובים עבור המודל KNN

```
parameters = {
    'metric': ['euclidean', 'manhattan', 'minkowski'],
    'algorithm': ['ball_tree', 'kd_tree', 'brute'],
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform'] }
gscv2 = GridSearchCV(KNeighborsClassifier(), parameters, cv=5)
gscv2.fit(X_train, y_train)
print(gscv2.best_params_)

{'algorithm': 'ball_tree', 'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'uniform'}
```

בניתי ואימנתי את המודל KNN לפי הפרמטרים שנבחרו, ובדקתי את הדיוק שלו

```
knn1 = KNeighborsClassifier(
    metric = gscv2.best_params_['metric'],
    algorithm = gscv2.best_params_['algorithm'],
    n_neighbors = gscv2.best_params_['n_neighbors'],
    weights = gscv2.best_params_['weights'])

knn1.fit(X_train, y_train)
score = knn1.score(X_test, y_test)

print(f"score: {score:.4f}")

0.9796
```

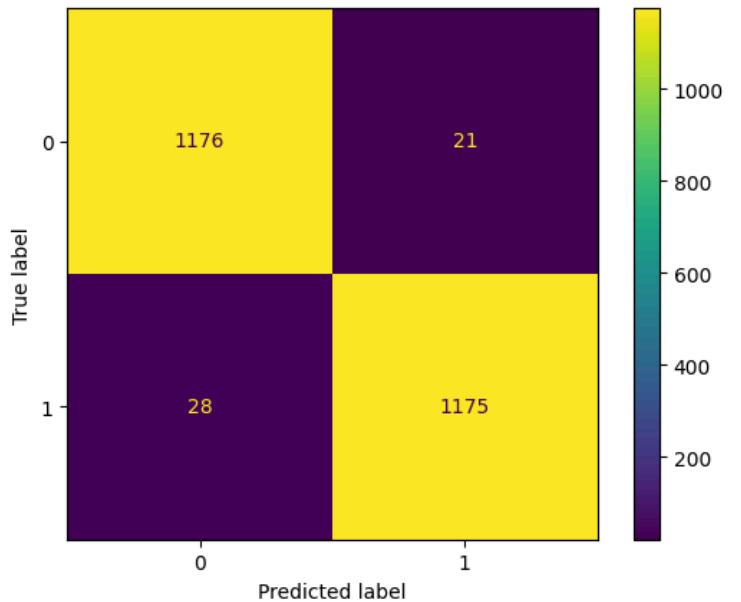
לאחר מכן בניתי ומודל עם הפרמטרים אשר מצאתי

```
cm = confusion_matrix(y_test, knn1.predict(X_test))
print(cm)

[[1176  21]
 [ 28 1175]]
```

הציגי מטריצת בלבול כדי לראות אם המודל טועה

```
disp = ConfusionMatrixDisplay(cm)
disp.plot()
plt.show()
```



ולבסוף השתמשתי ב `classification_report` כדי לחשב את דיוק המודל שלי

```
knn_pred = knn1.predict(X_test)

print(classification_report(y_test, knn_pred, digits=4))
```

	precision	recall	f1-score	support
0	0.9767	0.9825	0.9796	1197
1	0.9824	0.9767	0.9796	1203
accuracy			0.9796	2400
macro avg	0.9796	0.9796	0.9796	2400
weighted avg	0.9796	0.9796	0.9796	2400

## סיכום

השתמשתי בשני אלגוריתמים כדי לנבא את איות הבננות: רגרסיה לוגיסטיות ו-KNN (K-Neighbors). הדיק של הרגרסיה הלוגיסטי בסט הבדיקה היה 88%, והדיק של KNN היה 98%

זה מראה ש-KNN הצליח יותר במשימה, נראה כי הוא מתאים יותר לחלוקה מורכבת של הנתונים. אך, במקרה זה, האלגוריתם של KNN היה היעיל ביותר

## למה יש הבדל גדול בדיק בין המודלים?

רגרסיה לוגיסטי פועלת בצורה טוביה יותר כאשר הנתונים ניתנים לחלוקה בקי ישר. במקרה שלנו, הנתונים לא כל כך ברורים ומעורבים, מה שמקשה על המודול לסוג את הבננות בצורה נכונה. רגרסיה לוגיסטי מנסה למקום קו אחד שמחלק את שתי הקבוצות (בננות טובות ורעות), אבל בגלל שהנתונים מעורבים, המודול לאצליח לחלק אותן בצורה מדויקת

לעומת זאת, KNN עובד בצורה שונה. במקרה לחפש קו חיתוך, KNN בודק את השכנים הקרובים ביותר של כל נקודה (במקרה שלנו, כל בינה) ומחליט לאיזו קבוצה להקצתו אותה בהתאם על כך. גישה זו מאפשרת ל-KNN לקחת בחשבון יותר פרטים ולבצע עבודה טוביה יותר כשהנתונים מעורבים

לסיכום, בעוד שרגרסיה לוגיסטיות יכולה להיות יעילה כאשר יש חלוקה ברורה בין הקבוצות, במקרה שלנו, שבו הנתונים מעורבים ואין חלוקה ברורה, KNN מציגה תוצאות טובות יותר

## רפלקציה אישית

במהלך העבודה על הפרויקט למדתי איך לבנות מודלים של מידת מכונה ולהשווות ביניהם. הבנתי כמה חשוב להבין את מבנה הנתונים לפני שמאנים את המודל. בהתחלה היה לי קשה עם הקוד, אבל עם הזמן והתרגול, הצלחתי להבין את השלבים ולבסוף בצורה מסודרת. למדתי גם איך להסביר תוצאות בצורה פשוטה ולכתוב מסקנות מתוך הנתונים. בנוסף, השתמשתי בכלים של ויזואלייזציה כמו גרפים ומטריצת בלבול שעזרו לי להבין טוב יותר את הביצועים של המודלים וההתפלגות של הנתונים.

