Confronto

April 24, 2024

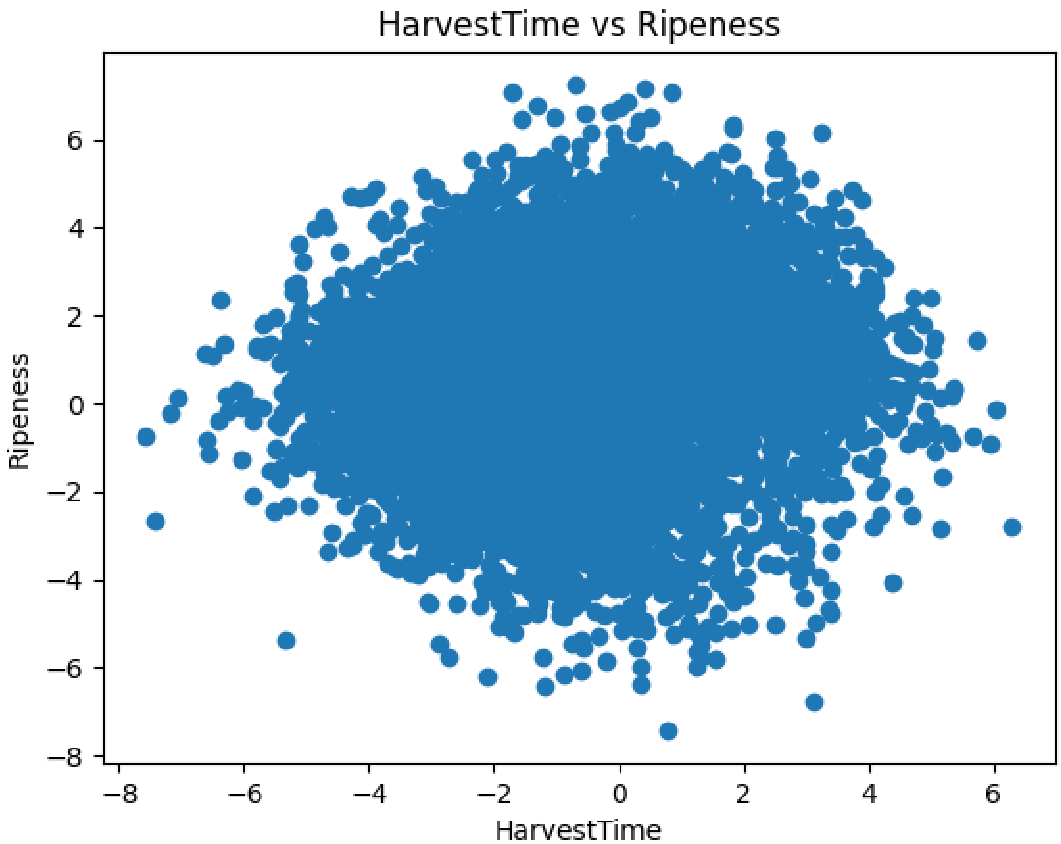
[ ]:
```python
import pandas as pd

df = pd.read_csv("banana_quality.csv")
```

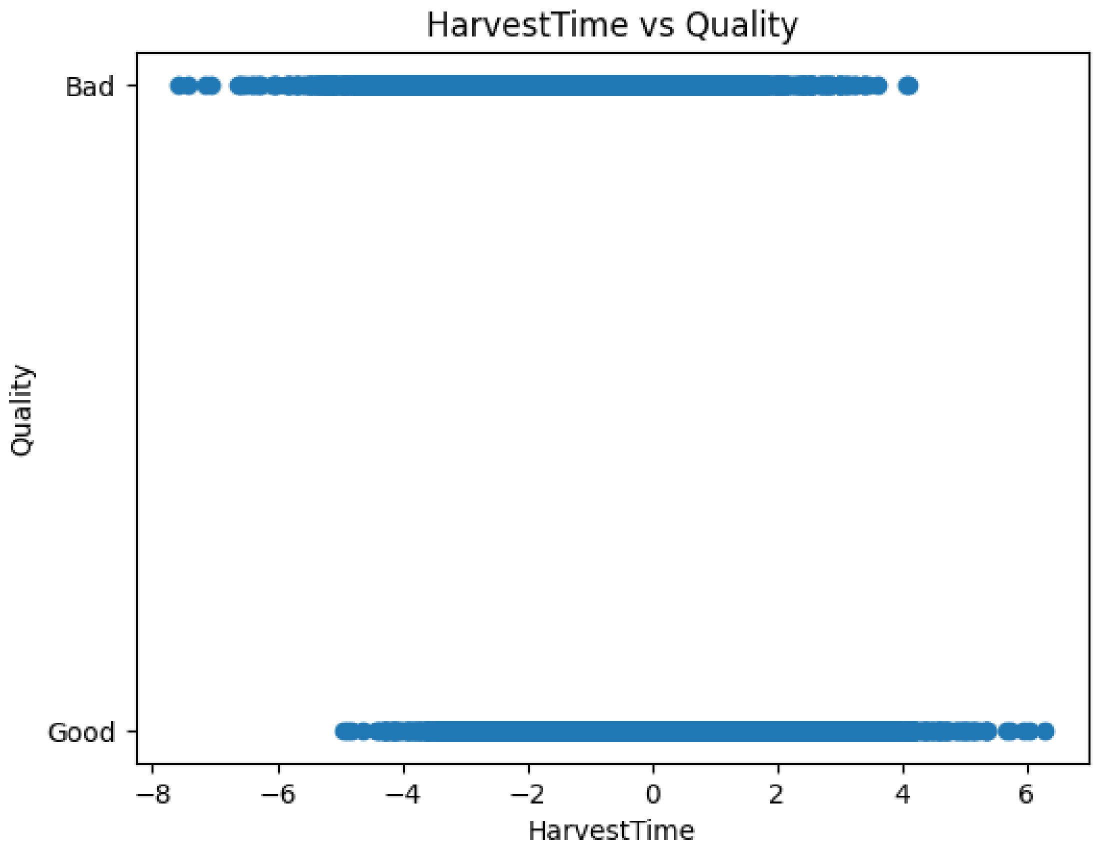
[ ]:
```python
df
```

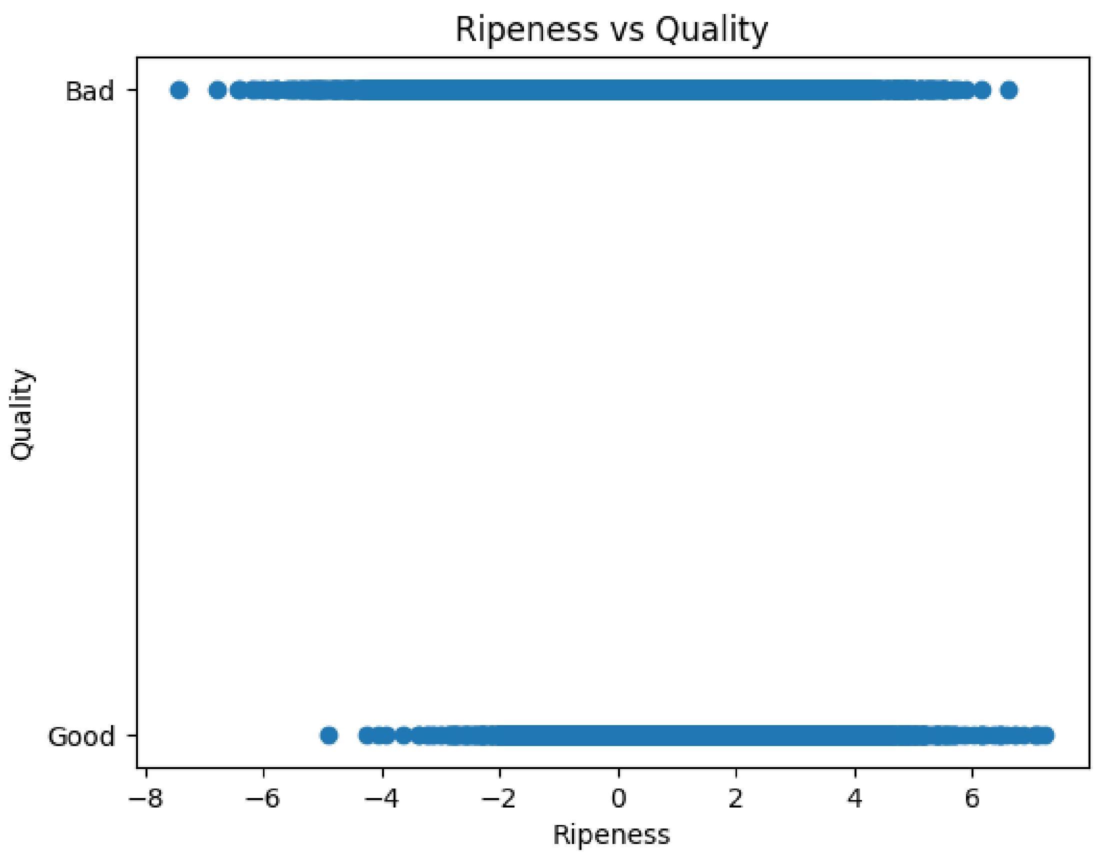
[ ]:
```
          Size    Weight  Sweetness  Softness  HarvestTime  Ripeness  \
0    -1.924968  0.468078   3.077832 -1.472177     0.294799  2.435570
1    -2.409751  0.486870   0.346921 -2.495099    -0.892213  2.067549
2    -0.357607  1.483176   1.568452 -2.645145    -0.647267  3.090643
3    -0.868524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

       Acidity Quality
0     0.271290    Good
1     0.307325    Good
2     1.427322    Good
3     0.477862    Good
4     2.812442    Good
…          …     …
7995  2.398091     Bad
7996  2.113136     Bad
7997  0.697361     Bad
7998  0.423569     Bad
7999  1.435644     Bad

[8000 rows x 8 columns]
```

[ ]:
```python
import matplotlib.pyplot as plt

harvest_time_values = df['HarvestTime']
ripeness_values = df['Ripeness']
```

```python
quality_values = df['Quality']

# Scatter plot HarvestTime vs Ripeness
plt.figure()
plt.scatter(harvest_time_values, ripeness_values)
plt.title('HarvestTime vs Ripeness')
plt.xlabel('HarvestTime')
plt.ylabel('Ripeness')
plt.show()

# Scatter plot HarvestTime vs Quality
plt.figure()
plt.scatter(harvest_time_values, quality_values)
plt.title('HarvestTime vs Quality')
plt.xlabel('HarvestTime')
plt.ylabel('Quality')
plt.show()

# Scatter plot Ripeness vs Quality
plt.figure()
plt.scatter(ripeness_values, quality_values)
plt.title('Ripeness vs Quality')
plt.xlabel('Ripeness')
plt.ylabel('Quality')
plt.show()
```


HarvestTime vs Ripeness
Ripeness
HarvestTime
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6


HarvestTime vs Quality
Bad
Good
Quality
−8
−6
−4
−2
0
2
4
6
HarvestTime




```
[ ]: import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

numeric_cols = df.select_dtypes(include=['number']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Define the transformations for numeric and categorical columns
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Combine the transformations using ColumnTransformer
preprocessor = ColumnTransformer(
```

```
    transformers=[
        ('num', numeric_transformer, numeric_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

# Fit and transform the data
df_es = pd.DataFrame(preprocessor.fit_transform(df))

df_es
```

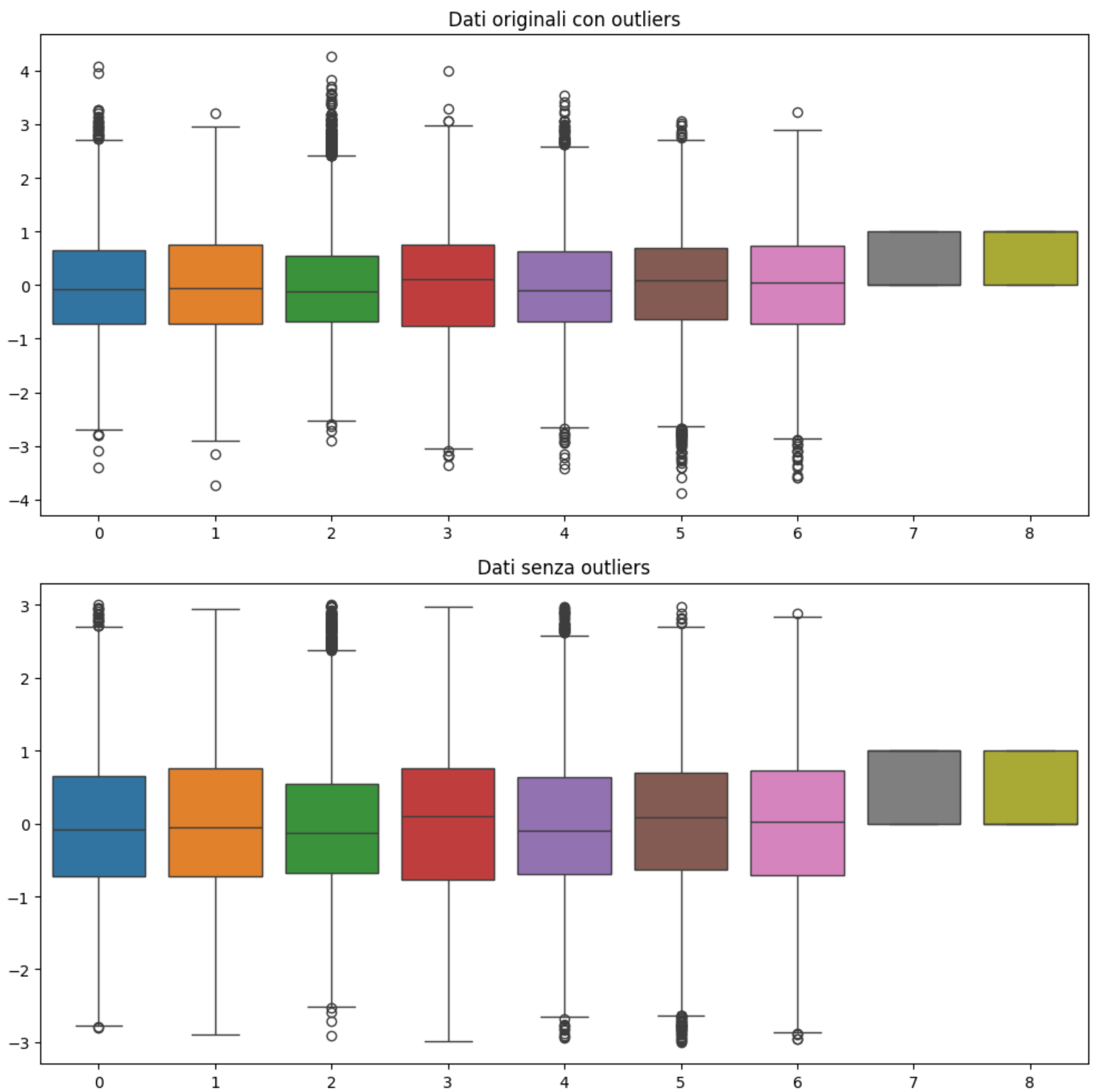
[ ]:

```
             0         1         2         3         4         5         6  \
0    -0.551136  0.609729  1.975051 -0.705896  0.523951  0.782568  0.114491
1    -0.778107  0.619052  0.573385 -1.201237 -0.070585  0.608493  0.130204
2     0.182685  1.113298  1.200347 -1.273895  0.052101  1.092419  0.618577
3    -0.056521  1.154485  1.365182 -0.609815 -0.127716  0.516472  0.204566
4     0.655290  1.031953  0.383797 -0.578798 -0.340291  0.140598  1.222556
…          …         …         …         …         …         …         …
7995 -2.653041  0.736471  0.977850  1.436842  0.525518 -0.443697  1.041879
7996  0.748609 -0.722715 -1.048050  0.243907 -0.286557 -1.465351  0.917625
7997  1.016216 -0.568827 -0.904435  0.474274  0.094620 -1.237308  0.300279
7998 -0.648022 -0.983020 -0.122055  1.036948 -0.025716 -2.062933  0.180892
7999 -0.895682 -0.636790  0.476946  0.733212 -0.416005 -1.129035  0.622206

        7    8
0     0.0  1.0
1     0.0  1.0
2     0.0  1.0
3     0.0  1.0
4     0.0  1.0
…     …    …
7995  1.0  0.0
7996  1.0  0.0
7997  1.0  0.0
7998  1.0  0.0
7999  1.0  0.0

[8000 rows x 9 columns]
```

[ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

z_scores = np.abs((df_es - df_es.mean()) / df_es.std())
threshold = 3
df_no_outliers = df_es[(z_scores < threshold).all(axis=1)]
fig, axes = plt.subplots(2, 1, figsize=(10, 10))
```

```python
sns.boxplot(data=df_es, ax=axes[0])
axes[0].set_title('Dati originali con outliers')
sns.boxplot(data=df_no_outliers, ax=axes[1])
axes[1].set_title('Dati senza outliers')

plt.tight_layout()
plt.show()
```



[ ]:
```python
from sklearn.model_selection import train_test_split

# Suddividiamo il dataset in set di addestramento e di test
train_df, test_df = train_test_split(df_no_outliers, test_size=0.2, random_state=42)
```

```
# Stampiamo le dimensioni dei set di addestramento e di test
print("Dimensioni del set di addestramento:", train_df.shape)
print("Dimensioni del set di test:", test_df.shape)
```

```
Dimensioni del set di addestramento: (6316, 9)
Dimensioni del set di test: (1580, 9)
```

[ ]:
```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
# Import other models you want to compare

# Step 1: Split the Data
X_train = train_df.drop(columns=[8])
y_train = train_df[8]

# Step 2: Choose Models
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(),
    # Add other models here
}

# Step 3: Define K-Fold Cross-Validation
k_folds = 5
kf = KFold(n_splits=k_folds, shuffle=True, random_state=42)

# Step 4 & 5: Train Models with Cross-Validation and Compare Performance
for model_name, model in models.items():
    scores = cross_val_score(model, X_train, y_train, cv=kf,␣
 ↪scoring='neg_mean_squared_error')
    # Use appropriate scoring metric based on your problem
    print(f'{model_name} Mean MSE: {scores.mean()}, Std: {scores.std()}')

# You can also choose a different performance metric and compare models based␣
 ↪on that metric
```

```
Linear Regression Mean MSE: -4.686933029116444e-31, Std: 4.087671435583819e-31
Decision Tree Mean MSE: 0.0, Std: 0.0
```

[ ]:
```

```