Dataset Documentation

Link to Dataset Documentation:

https://archive.ics.uci.edu/dataset/94/spambase

https://archive.ics.uci.edu/dataset/186/wine+quality

Linear Regression Analysis

Business Understanding / Objective

This project attempts to develop a predictive model to classify Portuguese "Vinho Verde" red wine quality based on some physicochemical properties including acidity, alcohol content, and sugar levels. By doing this we can predict a wine's sensory quality score based on these chemical properties, allowing wine producers and distributors to estimate quality further upstream in the production chain.

This dataset permits us to abstract the problem as a regression one, since in our case the output (to be predicted) is a continuous quality score. Such a model may provide producers with better decision making on what wines may perform better in a score (or possibly identifying areas in the process where quality may be improved).

Data Understanding / Exploration

Dataset Characteristics:
- Number of features: 11
- Target variable: Quality (0 to 10)
- Feature descriptions: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol

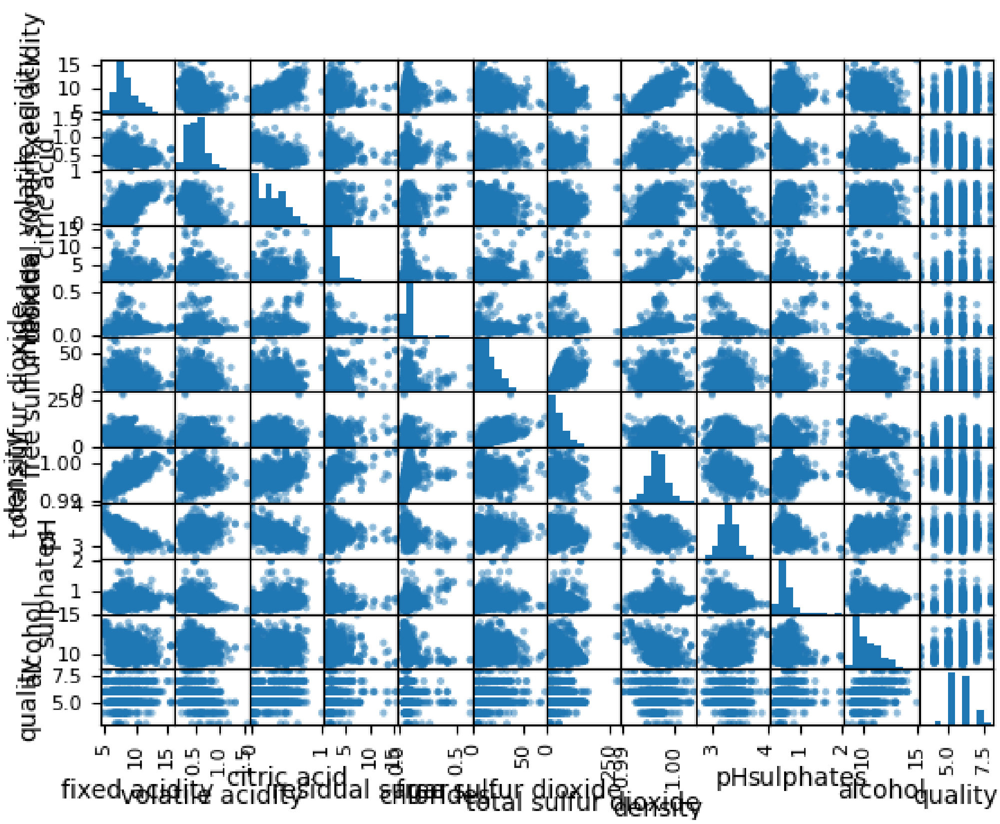
Code snippet for loading and exploring data:

```
df = pd.read_csv('data/winequality-red.csv', sep=';')
df.head()
df.describe()
df.corr()
scatter_matrix(df)
plt.show()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000



Based on the scatter matrix plotted, it is unlikely to get a good model because very little correlation can be seen between features and the target variable.

Modeling

The Linear Regression model was configured with a random_state for training/testing data split.

Code snippet for model fitting:

```
X = df.drop(columns = 'quality')

y = df.quality

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=1)

model = LinearRegression()
model.fit(X_train, y_train)
```

Evaluation

Evaluation metrics include R^2 score and root mean squared error (RMSE):

```
print('coefficient of determination R^2:', model.score(X_train, y_train))
yhat = model.predict(X_test)
print("RMSE:", mean_squared_error(y_test, yhat, squared=False))
```

coefficient of determination R^2: 0.36558499214790485

RMSE: 0.6189280908000773

Conclusion

The Linear Regression model provided an RMSE of 0.62 and an R^2 of 0.37. This analysis indicates that the model is bad for predicting the target variable in this business context since it performs poorly with low- and high-quality wines.

Decision Tree Analysis

Business Understanding / Objective

This project aims to build a predictive model to classify incoming emails as either "spam" or "not spam." This model could also be useful for people or businesses who want to filter out spam and save space in their inboxes and unwanted links.

This dataset utilizes the patterns that can be found within a spam or non-spam email, to determine the category of an email. For instance, non-spam emails may contain words (like "george" and area code "650") that are absent in spam emails, and spam emails contain regularities of non-personal computer usage (like particular phrase patterns typical of advertising or scams). Then we will use this knowledge of how language works to build a custom spam filter which will filter the mails very well and enhances user efficiency and email system.

Data Understanding / Exploration

Dataset Characteristics:
- Number of features: 57
- Target variable: Class (Spam or not spam)
- Feature descriptions: Frequency of characters appearing in an email

Code snippet for loading and exploring data:

```
col_names = [all features]
df = pd.read_csv('data/spambase.csv', header=None, names=col_names)
df.head()
df.describe()
```

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	word_freq_order	word_freq_mail	...	char_freq_;	char_fre
0	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00	0.00	...	0.00	0.
1	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94	...	0.00	0.
2	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25	...	0.01	0.
3	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.00	0.
4	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63	...	0.00	0.

5 rows × 58 columns

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	word_freq_order	word_freq_mail	...	char_freq_;
count	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	4601.000000	...	4601.000000
mean	0.104553	0.213015	0.280656	0.065425	0.312223	0.095901	0.114208	0.105295	0.090067	0.239413	...	0.038575
std	0.305358	1.290575	0.504143	1.395151	0.672513	0.273824	0.391441	0.401071	0.278616	0.644755	...	0.243471
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
75%	0.000000	0.000000	0.420000	0.000000	0.380000	0.000000	0.000000	0.000000	0.000000	0.160000	...	0.000000
max	4.540000	14.280000	5.100000	42.810000	10.000000	5.880000	7.270000	11.110000	5.260000	18.180000	...	4.385000

8 rows × 58 columns

Based on frequency charts that were plotted, words like 'you' and 'will' seems to appear a lot more.

Modeling

The model configuration includes setting a random_state for reproducibility and using train_test_split for training and test data separation.

Code snippet for model fitting:

```
X = df.drop(columns = 'Class')

y = df["Class"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = DecisionTreeClassifier()

model.fit(X_train, y_train)
```

Evaluation

Evaluation metrics include accuracy and confusion matrix:

```
yhat = model.predict(X_test)
accuracy = accuracy_score(y_test, yhat)

conf_matrix = confusion_matrix(y_test, yhat)

print('Accuracy:', accuracy)

print('Confusion Matrix:', conf_matrix)
```

```
Accuracy: 0.9077090119435396

Confusion Matrix: [[494  37]

 [ 48 342]]
```

Cross validation was needed for this decision tree since we need to find the best depth for the model.

Finding the best depth for the decision tree using cross validation:

```
for d in range(2, 20):

   model = DecisionTreeClassifier(max_depth=d)

   scores = cross_val_score(model, X_train, y_train, cv=5)

   print(f"Depth: {d}, Mean Cross-Validation Score: {scores.mean():.4f}")
```

```
Depth: 2, Mean Cross-Validation Score: 0.8677
```

Depth: 3, Mean Cross-Validation Score: 0.8785

Depth: 4, Mean Cross-Validation Score: 0.9016

Depth: 5, Mean Cross-Validation Score: 0.9060

Depth: 6, Mean Cross-Validation Score: 0.9109

Depth: 7, Mean Cross-Validation Score: 0.9144

Depth: 8, Mean Cross-Validation Score: 0.9163

Depth: 9, Mean Cross-Validation Score: 0.9204

Depth: 10, Mean Cross-Validation Score: 0.9136

Depth: 11, Mean Cross-Validation Score: 0.9174

Depth: 12, Mean Cross-Validation Score: 0.9171

Depth: 13, Mean Cross-Validation Score: 0.9185

Depth: 14, Mean Cross-Validation Score: 0.9160

Depth: 15, Mean Cross-Validation Score: 0.9117

Depth: 16, Mean Cross-Validation Score: 0.9144

Depth: 17, Mean Cross-Validation Score: 0.9166

Depth: 18, Mean Cross-Validation Score: 0.9163

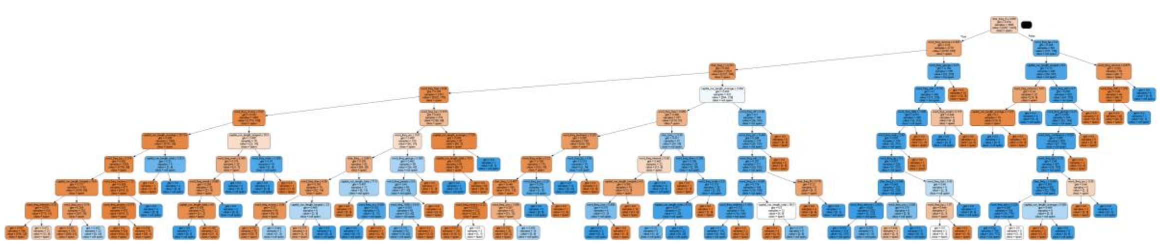
Depth: 19, Mean Cross-Validation Score: 0.9136

By using cross validation, the decision tree with a max depth of 9 appears to have the highest accuracy.

Retrain the model with the best decision tree (depth = 9) and creating a plot for it:

```
model = DecisionTreeClassifier(max_depth = 9)

model.fit(X_train, y_train)

# Predicting the target on the test set

yhat = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, yhat)

conf_matrix = confusion_matrix(y_test, yhat)


print('Accuracy:', accuracy)

print('Confusion Matrix:', conf_matrix)

Accuracy: 0.9218241042345277

Confusion Matrix: [[508  23]

 [ 49 341]]

target_names = ["spam", "not spam"]

dot_data = StringIO()

export_graphviz(model, out_file=dot_data,

 filled=True, rounded=True,

 special_characters=True, feature_names = features,

 class_names = target_names)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

graph.write_png('plots/spambase_bestDepth.png')

Image(graph.create_png())
```



Conclusion

The Decision Tree model performed with an accuracy of 92%, indicating it is effectively distinguishing between spam and non-spam emails, making it a strong model for this classification task.