What makes a great wine?

A machine learning approach to a very old question

Mitali Shroff and Thomas Dohle
CS 6620 Northeastern University

The questions

1. How can we predict the quality of a wine without tasting it?
2. How can we predict if a wine is red or white without looking at it?

Motivation

Wine sales generated ~$75 billion in the United states in 2019[(1)]. Good wine is a lucrative product!

1. https://wineinstitute.org/our-industry/statistics/california-us-wine-sales/

The data

Source: UC Irvine Machine Learning Repository

Publication:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009

Red wine: 1599 entries

White wine: 4898 entries

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
3918	6.4	0.35	0.28	1.6	0.037	31.0	113.0	0.98779	3.12	0.40	14.20	7
4503	5.8	0.61	0.01	8.4	0.041	31.0	104.0	0.99090	3.26	0.72	14.05	7
652	15.9	0.36	0.65	7.5	0.096	22.0	71.0	0.99760	2.98	0.84	14.90	5

Question 1: Predicting quality

- Regression models were created for 3 unique datasets: red wine only, white wine only, red and white wine together
- Three different methods of feature selection were used
  - One feature at a time
  - All features at once
  - A “smart” method of feature selection that tried all combinations of features with an absolute value correlation >=.1

```python
def generate_regression_data(wine_type='red', features=None):
    if wine_type == 'red':
        data = red_wine_data
    elif wine_type == 'both':
        data = all_data
    elif wine_type == 'white':
        data = white_wine_data
    else:
        raise Exception ('Invalid selection for wine_type')
    y = data['quality']
    # feature=None corresponds to calculating data for all features
    if features:
        X = data[features]
    else:
        X = data.drop(columns='quality')

    x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=.8, test_size=.2)
    reg = LinearRegression().fit(x_train,y_train)
    prediction = reg.predict(x_test)
    coefficients = reg.coef_
    mse = np.mean((prediction, y_test))
    variance = reg.score(x_test, y_test)
    data = dict(
        feature=X,
        regression=reg,
        prediction=prediction,
        coefficients=coefficients,
        mse=mse,
        variance=variance,
        x_test=x_test,
        x_train=x_train,
        y_test=y_test,
        y_train=y_train,
    )
    return data
```

```python
import itertools
def find_highest_variance_regression(wine_type, features):
    """
    Given a list of strings representing the column names of features, genereate regression models for
    all possible combinations of features.
    """
    combos = []
    data_dict = {}
    for l in range(0, len(features)+1):
        for subset in itertools.combinations(both_features, l):
            combos.append(subset)
    for combo in combos:
        data = generate_regression_data(wine_type=wine_type, features=list(combo))
        data_dict[combo] = data
    max_key = max(data_dict, key=lambda tup: data_dict[tup]['variance'])
    return data_dict[max_key], max_key
```

```
alcohol                 0.476166
sulphates               0.251397
citric acid             0.226373
fixed acidity           0.124052
residual sugar          0.013732
free sulfur dioxide    -0.050656
pH                     -0.057731
chlorides              -0.128907
density                -0.174919
total sulfur dioxide   -0.185100
volatile acidity       -0.390558
Name: quality, dtype: float64
```

Correlation values for red wine

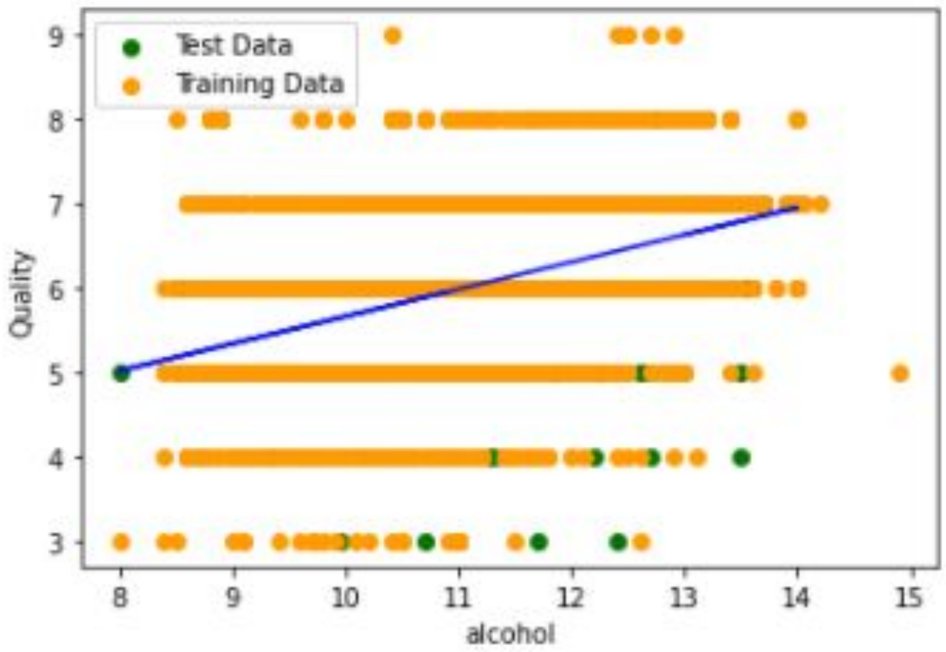
And the winner is...

Red wine only: all 11 features (variance = .389)

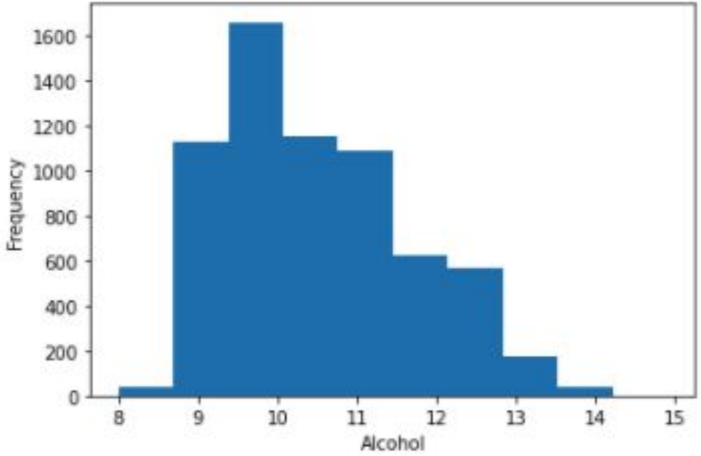
White wine only: volatile acidity, chlorides, density, alcohol (variance = .300)

Red and white wines: volatile acidity, chlorides, alcohol (variance = .304)

Red and White

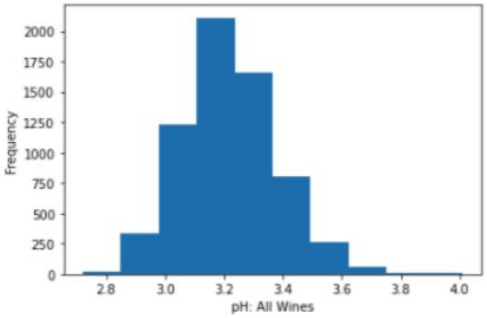


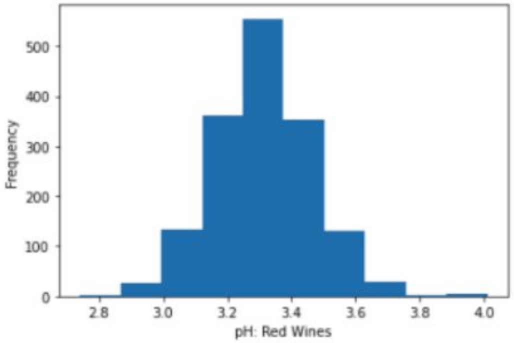
So should we make wine with 20, 30, or 40% alcohol?

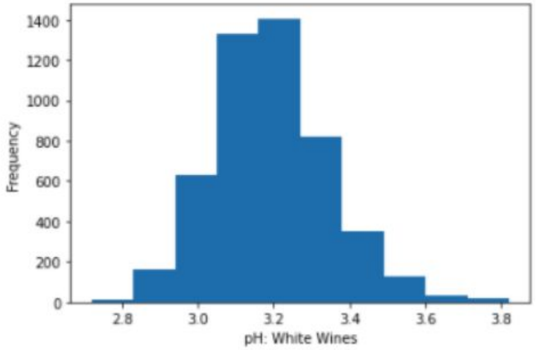


Mean quality score of wine with >13% alcohol content: 6.7
Mean quality score of all wines: 5.8

Other interesting observations



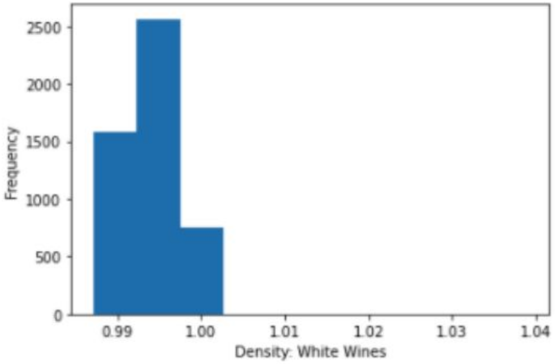


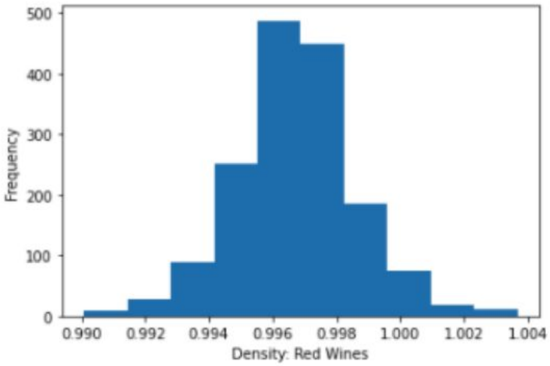


- pH didn't come up as particularly relevant
- Little difference in pH values of red vs white wines

Other interesting observations

- Density correlates negatively in all three datasets
- Most significant in white wine
  - Coefficient in white wine: -145
  - Coefficient in red wine: -11.0
- More uniform distribution in red wine





Future direction to improve these models

1. Validate with data from a different source
2. Obtain datasets of other types of wine (rose, prosecco, etc)
3. Deploy the models as a django or flask web app to allow users from non-cs backgrounds to utilize them

Question 2: Predicting type of wine

- Classification models were created on the entire wine dataset containing both red and white wine
- Feature selection techniques were used to identify which features were most important for this dataset
- Three different methods of classification were used
    - Decision Tree classification
    - Support Vector classification
    - K Nearest Neighbor classification

Method

1. Look for missing values and null check

```
print(wine_data.isnull().any())
```

```
fixed acidity           False
volatile acidity        False
citric acid             False
residual sugar          False
chlorides               False
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                      False
sulphates               False
alcohol                 False
quality                 False
type                    False
dtype: bool
```

```
wine_data.isna().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
type                    0
dtype: int64
```

Method

1. Look for missing values and null check

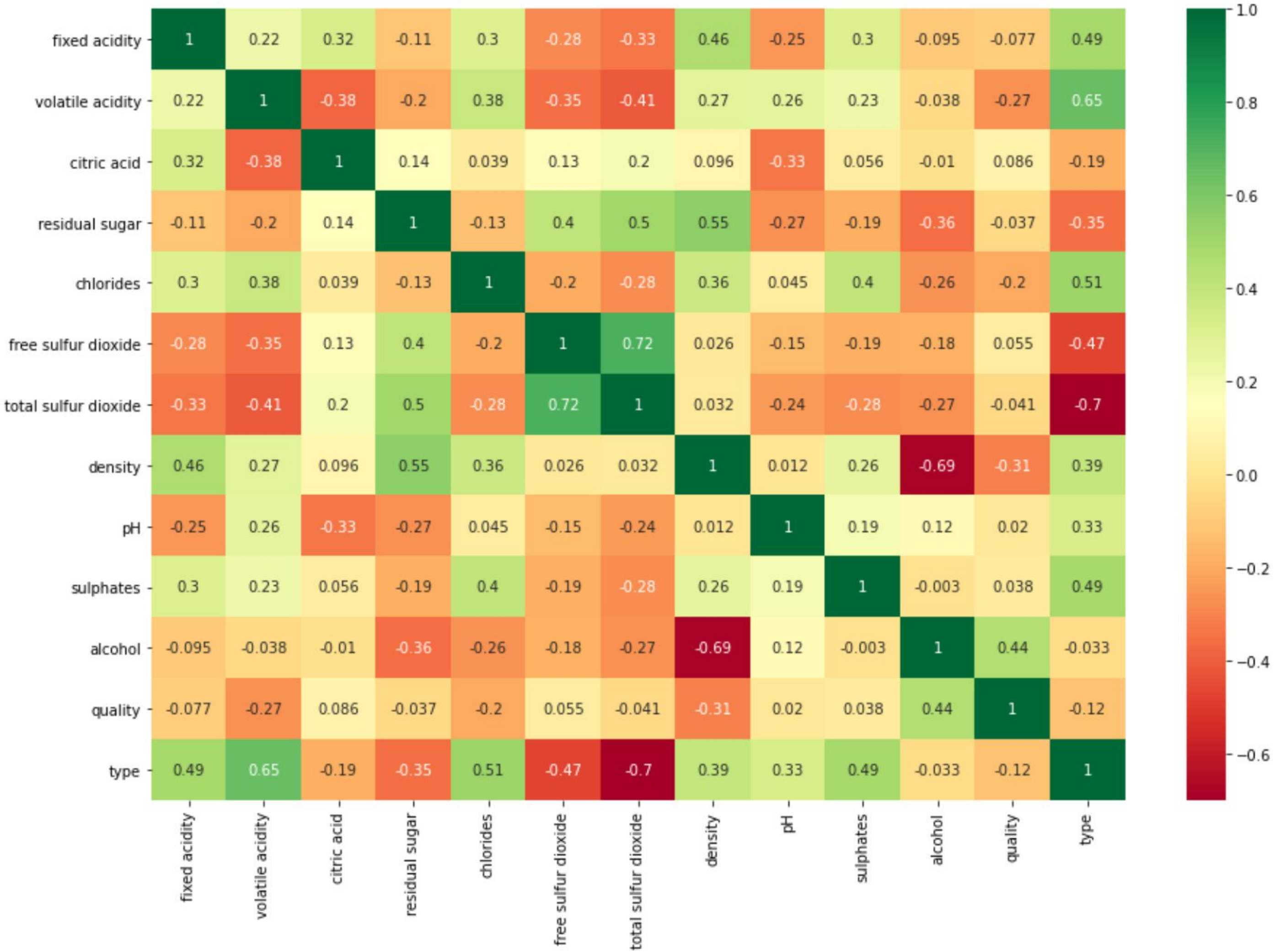
2. Add classification labels and create dataset

```
white_wine_data = pd.read_csv('winequality-white.csv', sep=';')
red_wine_data = pd.read_csv('winequality-red.csv', sep=';')

# Classify based on wine color, stored as the property "type". Here red=1, white=0
red_wine_data['type'] = 1
white_wine_data['type'] = 0

# Create the wine dataset
wine_data = pd.concat([red_wine_data, white_wine_data], sort=False)
# Shuffle it
wine_data = wine_data.sample(frac=1, random_state=101).reset_index(drop=True)
```

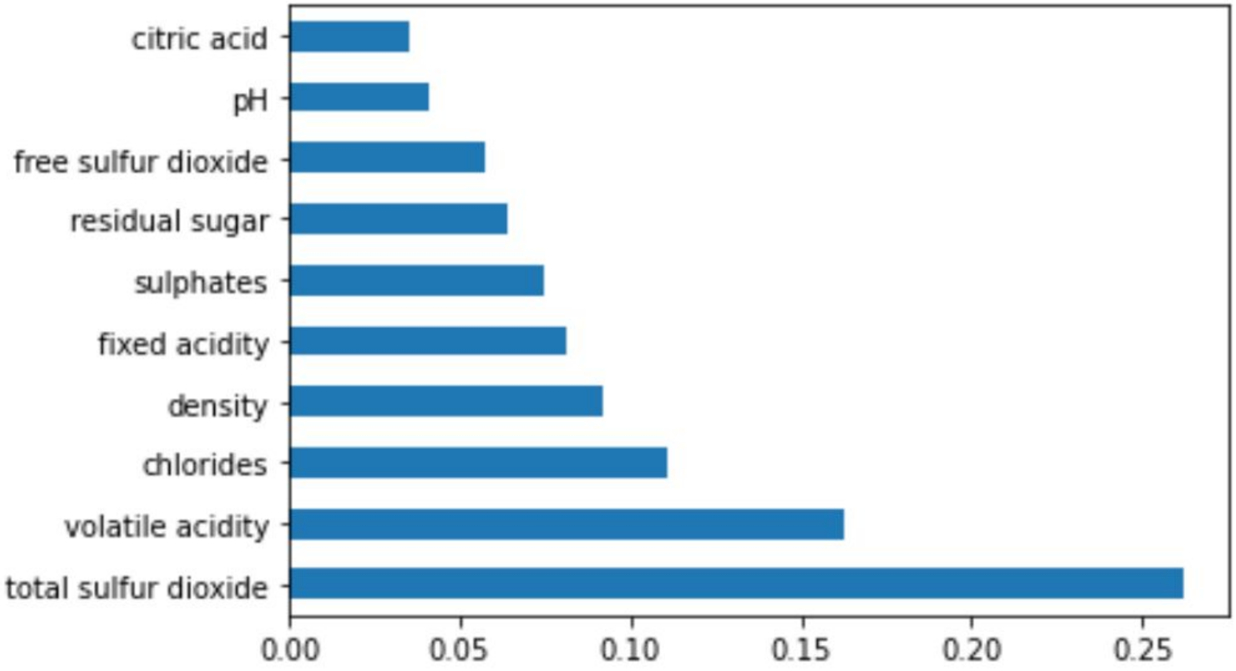
Method

1. Look for missing values and null check
2. Add classification labels and create dataset
3. Perform feature set analysis to obtain best feature set

Heatmap depicting Correlation between features



Extra-trees classifier



The following features seemed important to the dataset:

Correlation

- total sulfur dioxide
- volatile acidity
- chlorides
- fixed acidity
- sulphates

Extra-trees classifier

- total sulfur dioxide
- volatile acidity
- chlorides
- density

Method

1. Look for missing values and null check
2. Add classification labels and create dataset
3. Perform feature set analysis to obtain best feature set
4. Split dataset into training and test

```
def get_train_test_datasets(X, y, size):
    X_tr, X_t, y_tr, y_t = train_test_split(X, y, test_size = size, random_state = 0)
    return X_tr, X_t, y_tr, y_t
```

Method

1. Look for missing values and null check
2. Add classification labels and create dataset
3. Perform feature set analysis to obtain best feature set
4. Split dataset into training and test
5. Generate classification models and fit training dataset

Decision Tree Classification

```python
def decision_tree_classification(X, y, X_t, y_t):
    max_score = 0
    y_pred_dt_max = []
    for i in range(1, 19):
        dt_model = DecisionTreeClassifier(max_depth=i).fit(X, y)
        y_pred_dt = dt_model.predict(X_t)
        if accuracy_score(y_t, y_pred_dt) > max_score:
            max_score = accuracy_score(y_t, y_pred_dt)
            y_pred_dt_max = y_pred_dt

    print_results(y_t, y_pred_dt_max)
```

Support Vector Machine classification

```python
def svm_classification(X, y, X_t, y_t):
    svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X, y)
    y_pred_svm = svm_model_linear.predict(X_t)

    print_results(y_t, y_pred_svm)
```

K Nearest Neighbors Classification

```
def knn_classification(X, y, X_t, y_t):
    max_score = 0
    y_pred_knn_max = []
    for i in range(1,11):
        knn_model = KNeighborsClassifier(n_neighbors=i)
        knn_model.fit(X, y)
        y_pred_knn = knn_model.predict(X_t)
        if accuracy_score(y_t, y_pred_knn) > max_score:
            max_score = accuracy_score(y_t, y_pred_knn)
            y_pred_knn_max = y_pred_knn

    print_results(y_t, y_pred_knn_max)
```

Method

1. Look for missing values and null check
2. Add classification labels and create dataset
3. Perform feature set analysis to obtain best feature set
4. Split dataset into training and test
5. Generate classification models and fit training dataset
6. Predict the outcomes and tabulate accuracy metrics

Decision tree classification on full wine dataset

```
decision_tree_classification(X_train, y_train, X_test, y_test)
```

```
Confusion matrix:
 [[961   9]
 [ 17 313]]

Accuracy score:  0.98

Classification report:
               precision    recall  f1-score   support

           0       0.98      0.99      0.99       970
           1       0.97      0.95      0.96       330

    accuracy                           0.98      1300
   macro avg       0.98      0.97      0.97      1300
weighted avg       0.98      0.98      0.98      1300
```

SVM classification on full wine dataset

```
svm_classification(X_train, y_train, X_test, y_test)
```

```
Confusion matrix:
 [[967   3]
 [ 16 314]]

Accuracy score:  0.9853846153846154

Classification report:
               precision    recall  f1-score   support

           0       0.98      1.00      0.99       970
           1       0.99      0.95      0.97       330

    accuracy                           0.99      1300
   macro avg       0.99      0.97      0.98      1300
weighted avg       0.99      0.99      0.99      1300
```

KNN classification on full wine dataset

```
knn_classification(X_train, y_train, X_test, y_test)
```

```
Confusion matrix:
 [[945  25]
 [ 38 292]]

Accuracy score:  0.9515384615384616

Classification report:
               precision    recall  f1-score   support

           0       0.96      0.97      0.97       970
           1       0.92      0.88      0.90       330

    accuracy                           0.95      1300
   macro avg       0.94      0.93      0.94      1300
weighted avg       0.95      0.95      0.95      1300
```

Decision tree classification on subset of wine dataset

```
decision_tree_classification(X_train_subset, y_train_subset, X_test_subset, y_test_subset)
```

```
Confusion matrix:
 [[961   9]
 [ 14 316]]

Accuracy score:  0.9823076923076923

Classification report:
               precision    recall  f1-score   support

           0       0.99      0.99      0.99       970
           1       0.97      0.96      0.96       330

    accuracy                           0.98      1300
   macro avg       0.98      0.97      0.98      1300
weighted avg       0.98      0.98      0.98      1300
```

KNN classification on subset of wine dataset

```
knn_classification(X_train_subset, y_train_subset, X_test_subset, y_test_subset)
```

```
Confusion matrix:
 [[946  24]
 [ 44 286]]

Accuracy score:  0.9476923076923077

Classification report:
               precision    recall  f1-score   support

           0       0.96      0.98      0.97       970
           1       0.92      0.87      0.89       330

    accuracy                           0.95      1300
   macro avg       0.94      0.92      0.93      1300
weighted avg       0.95      0.95      0.95      1300
```

SVM classification on subset of wine dataset

```
svm_classification(X_train_subset, y_train_subset, X_test_subset, y_test_subset)
```

```
Confusion matrix:
 [[963   7]
 [ 21 309]]

Accuracy score:  0.9784615384615385

Classification report:
               precision    recall  f1-score   support

           0       0.98      0.99      0.99       970
           1       0.98      0.94      0.96       330

    accuracy                           0.98      1300
   macro avg       0.98      0.96      0.97      1300
weighted avg       0.98      0.98      0.98      1300
```

Support Vectore classification on scaled input (entire dataset)

```
svm_classification(X_train_scaled, y_train_scaled, X_test_scaled, y_test_scaled)
```

```
Confusion matrix:
 [[967   3]
 [  6 324]]

Accuracy score:  0.9930769230769231

Classification report:
               precision    recall  f1-score   support

           0       0.99      1.00      1.00       970
           1       0.99      0.98      0.99       330

    accuracy                           0.99      1300
   macro avg       0.99      0.99      0.99      1300
weighted avg       0.99      0.99      0.99      1300
```

KNN classification on scaled input (entire dataset)

```
knn_classification(X_train_scaled, y_train_scaled, X_test_scaled, y_test_scaled)
```

```
Confusion matrix:
 [[967   3]
 [  7 323]]

Accuracy score:  0.9923076923076923

Classification report:
               precision    recall  f1-score   support

           0       0.99      1.00      0.99       970
           1       0.99      0.98      0.98       330

    accuracy                           0.99      1300
   macro avg       0.99      0.99      0.99      1300
weighted avg       0.99      0.99      0.99      1300
```

Interesting observations

- We see that all the models perform well on this dataset (>90% accuracy)
- The best performing was the simple Linear Support Vector Machine on the entire dataset, followed by the Decision Tree classification model on the subset of important features (both give over 98% accuracy)
- Choosing a subset of important features via feature selection created comparably accurate models
- We used a standard scaler from sklearn and reached an accuracy of 99% for SVM and KNN on the entire dataset

Future direction to improve these models

1. Obtain datasets of other types of wine (rose, prosecco, etc)
2. Tweak the parameters of the classification models
3. Perform classification on another feature - quality

Helpful links

The code

The google site

Regression web app

Classification web app