

```
+ Code + Text
```

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
import time
import random

random.seed(100)
path="/content/drive/MyDrive/winequality-red.csv"
df=pd.read_csv(path)
df.head()
print('Dataset has'. df.shape[0] . 'rows and'. df.shape[1] . 'columns')
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
[21] import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
import time
import random

random.seed(100)
path="/content/drive/MyDrive/winequality-red.csv"
df=pd.read_csv(path)
df.head()
print('Dataset has', df.shape[0] , 'rows and', df.shape[1], 'columns')
```

<> Dataset has 1599 rows and 12 columns

[8] df.head()

Automatic saving failed. This file was updated remotely or in another tab. Show diff

acidity acidity acid sugar chlorides free sulfur dioxide total sulfur dioxide density pH sulphates alcohol quality

Dataset has 1599 rows and 12 columns

1s df.head()

	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

0s [9] df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
#   Column              Non-Null Count  Dtype
---
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   fixed acidity          1599 non-null   float64
 1   volatile acidity       1599 non-null   float64
 2   citric acid            1599 non-null   float64
 3   residual sugar         1599 non-null   float64
 4   chlorides              1599 non-null   float64
 5   free sulfur dioxide    1599 non-null   float64
 6   total sulfur dioxide   1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
10   alcohol               1599 non-null   float64
11   quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
[10] print(df.columns)
      print(df.shape)
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

+ Code + Text RAM Disk

```
print(df.columns)
print(df.shape)
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
      'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
      'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
(1599, 12)
```

```
[13]
df.duplicated().sum()

240
```

```
[12] df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.600000	7.10000	7.900000	9.200000	15.900000
volatile acidity	1599.0	0.527821	0.179060	0.120000	0.39000	0.520000	0.640000	1.580000
citric acid	1599.0	0.270976	0.194801	0.000000	0.09000	0.260000	0.420000	1.000000
chlorides	1599.0	0.087467	0.047065	0.012000	0.07000	0.070000	0.090000	0.611000

Automatic saving failed. This file was updated remotely or in another tab. Show diff

0s [12]

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.31113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.0000	6.00000	6.000000	8.00000

[14] df.nunique()

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
df.nunique()
```

fixed acidity	96
volatile acidity	143
citric acid	80
residual sugar	91
chlorides	153
free sulfur dioxide	60
total sulfur dioxide	144
density	436
pH	89
sulphates	96
alcohol	65
quality	6

dtype: int64

```
[23]
from sklearn.preprocessing import LabelEncoder
bins = (2, 6.5, 8)
group_names = ['bad', 'good']
df['quality'] = pd.cut(df['quality'], bins = bins, labels = group_names)
label_quality = LabelEncoder()
df['quality'] = label_quality.fit_transform(df['quality'])
df['quality'].value_counts()
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

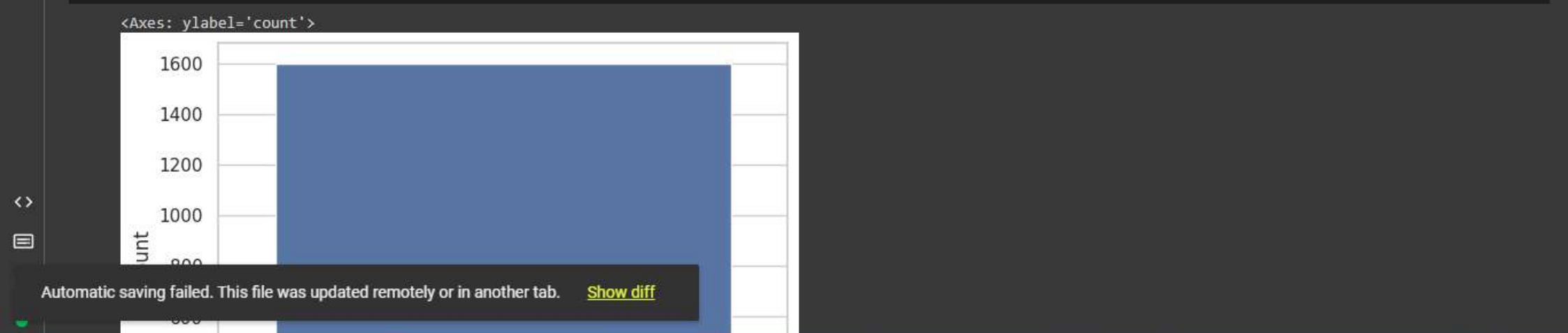
+ Code + Text

RAM Disk

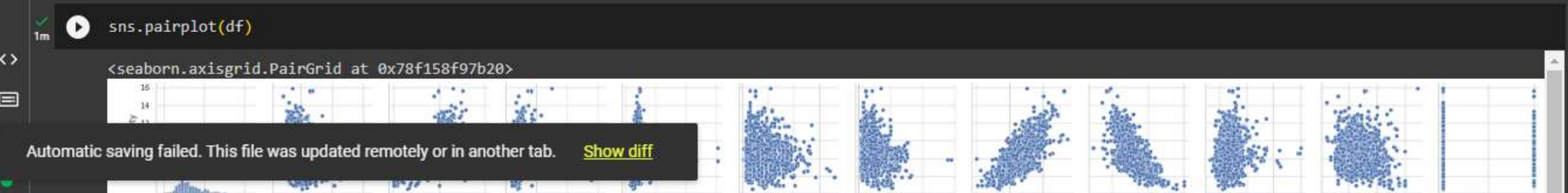
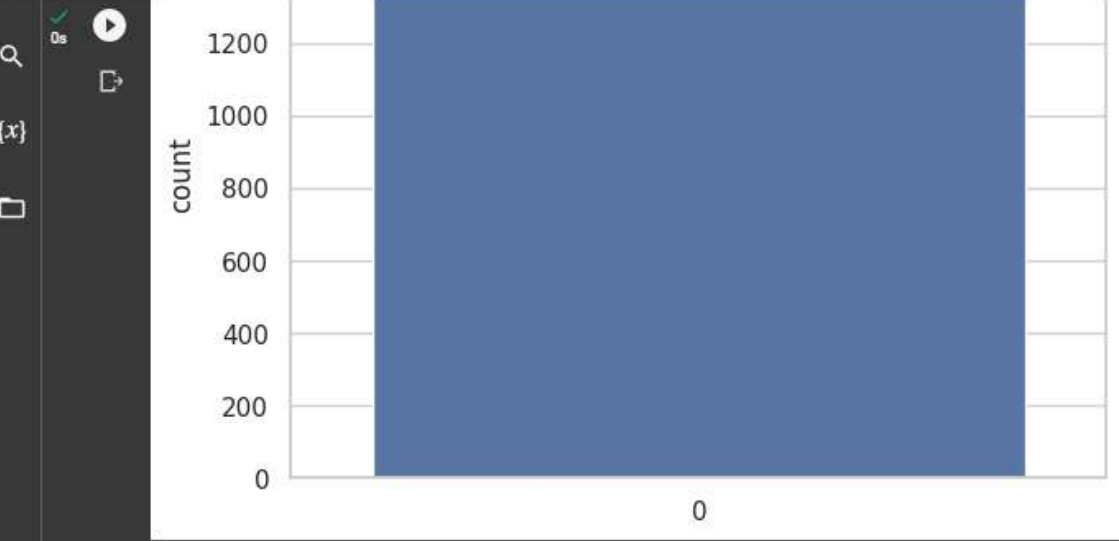
```
group_names = ['bad', 'good']
df['quality'] = pd.cut(df['quality'], bins = bins, labels = group_names)
label_quality = LabelEncoder()
df['quality'] = label_quality.fit_transform(df['quality'])
df['quality'].value_counts()
```

```
0    1382
1     217
Name: quality, dtype: int64
```

```
sns.countplot(df['quality'])
```

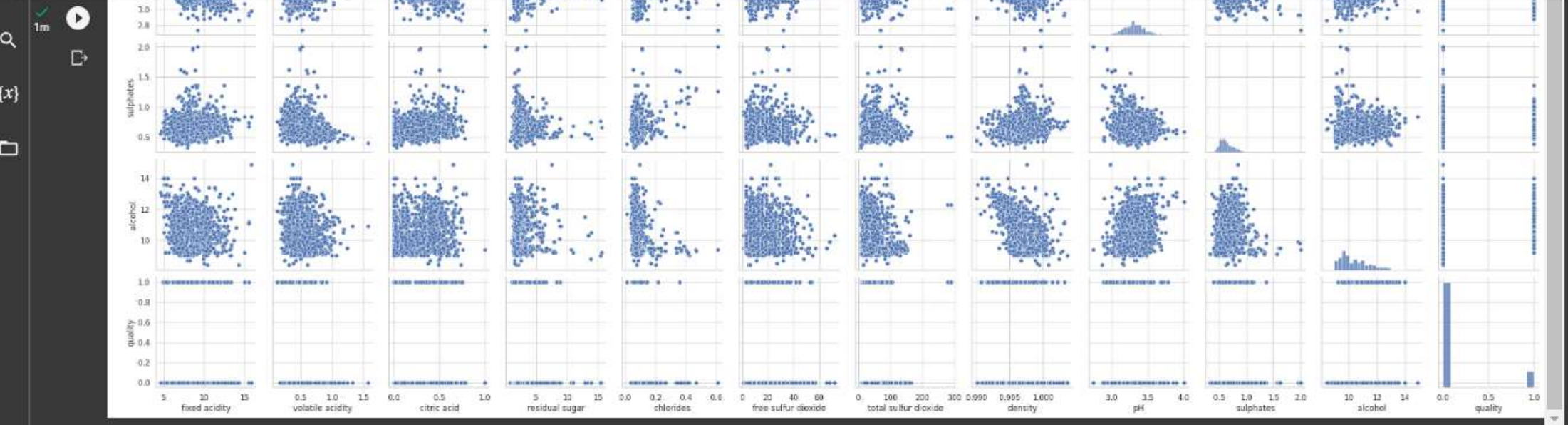


Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)









```
[26] fig = plt.figure(figsize=(15, 12))
plt.suptitle('Histograms of Numerical Columns', fontsize=20)
for i in range(df.shape[1]):
```

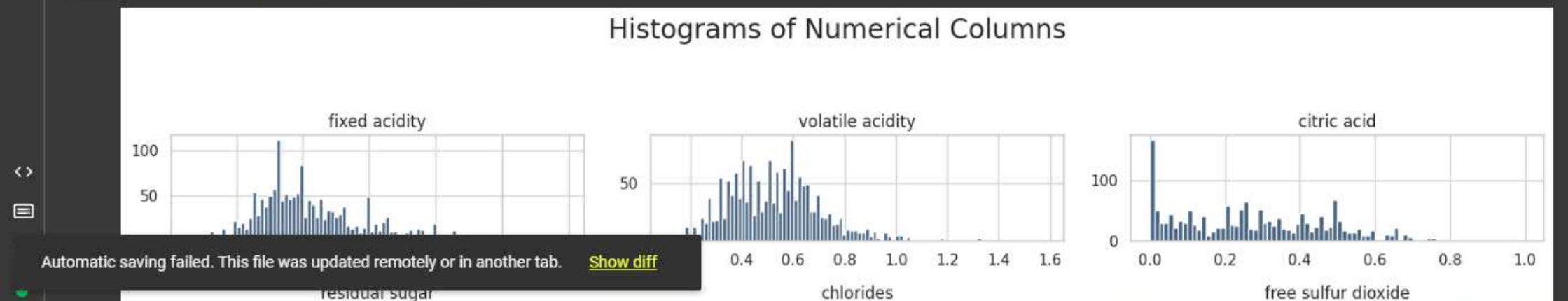
Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

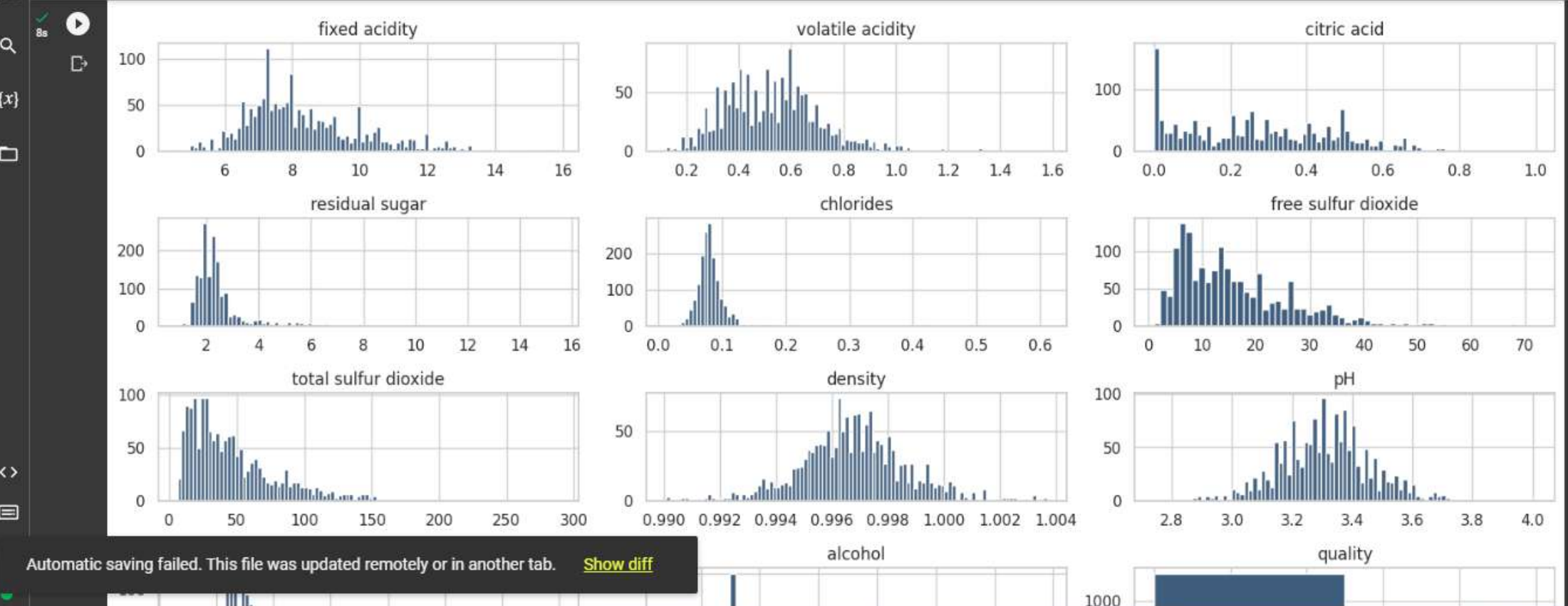

```
+ Code + Text
```

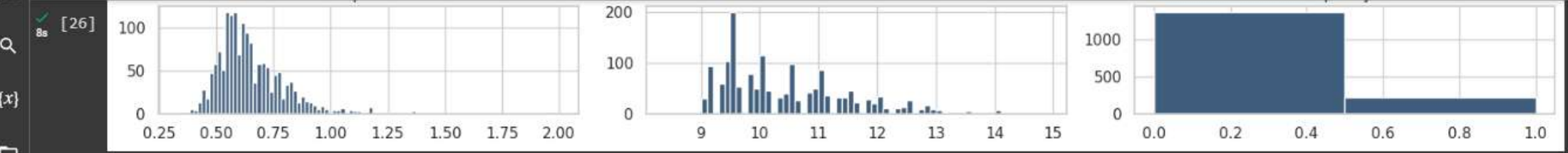
```
[26] fig = plt.figure(figsize=(15, 12))
plt.suptitle('Histograms of Numerical Columns', fontsize=20)
for i in range(df.shape[1]):
    plt.subplot(6, 3, i + 1)
    f = plt.gca()
    f.set_title(df.columns.values[i])

    vals = np.size(df.iloc[:, i].unique())
    if vals >= 100:
        vals = 100

    plt.hist(df.iloc[:, i], bins=vals, color='#3F5D7D')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```







```
[27] df.isna().any()

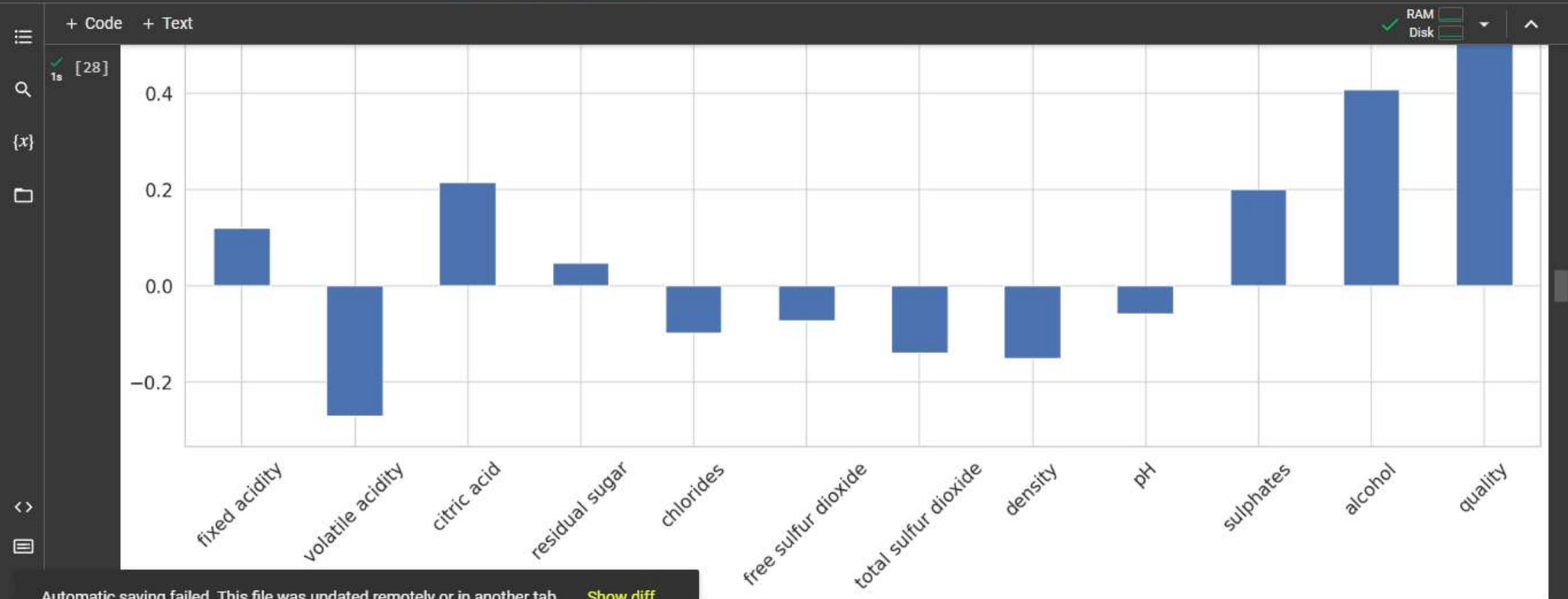
fixed acidity      False
volatile acidity   False
citric acid        False
residual sugar     False
chlorides          False
free sulfur dioxide False
total sulfur dioxide False
density            False
pH                False
sulphates          False
alcohol            False
quality            False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

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#) fontsize = 15,

```
+ Code + Text
```

```
[28] df.corrwith(df.quality).plot.bar(
      figsize = (20, 10), title = "Correlation with quality", fontsize = 15,
      rot = 45, grid = True)
```





Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

+ Code + Text

RAM Disk

1s [28]

0s

```
sns.set(style="white")
corr = df.corr()
corr.head()
```

	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	0.120061
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.270712
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.214716
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.047779
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.097308

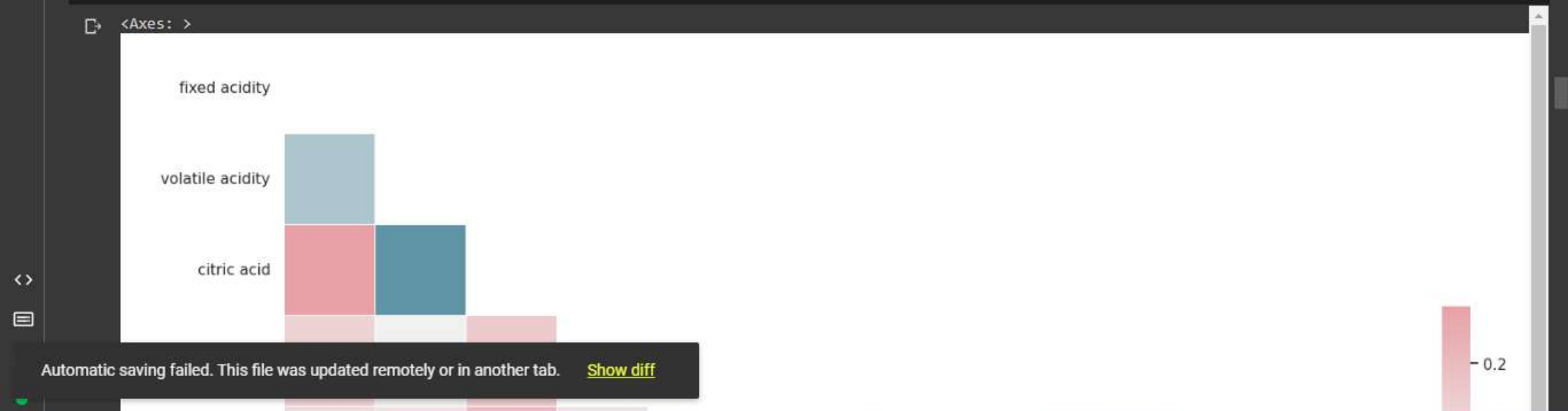
<>

1s

```
[31] mask = np.zeros_like(corr, dtype=np.bool)
      mask[np.triu_indices_from(mask)] = True
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(18, 15))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```



Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)





+ Code + Text

RAM Disk

```
[41] X = df.drop('quality',axis=1)
      y=df['quality']
      X.head()
      y.head()
      features_label=df.columns[:11]
```

```
X = df.drop('quality',axis=1)
y=df['quality']
X.head()
```

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

```
[35] v.head()
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
[35] y.head()
```

```
0 0
1 0
2 0
3 0
4 0
Name: quality, dtype: int64
```

```
[39] features_label=df.columns[:11]
```

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 200, criterion = 'entropy', random_state = 0)
classifier.fit(X, y)
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
for i in range(X.shape[1]):
    print ("%2d) %-*s %f" % (i + 1, 30, features_label[i],importances[indices[i]]))
```

```
1) fixed acidity      0.178690
2) volatile acidity  0.128748
3) citric acid       0.106496
4) residual sugar    0.095718
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

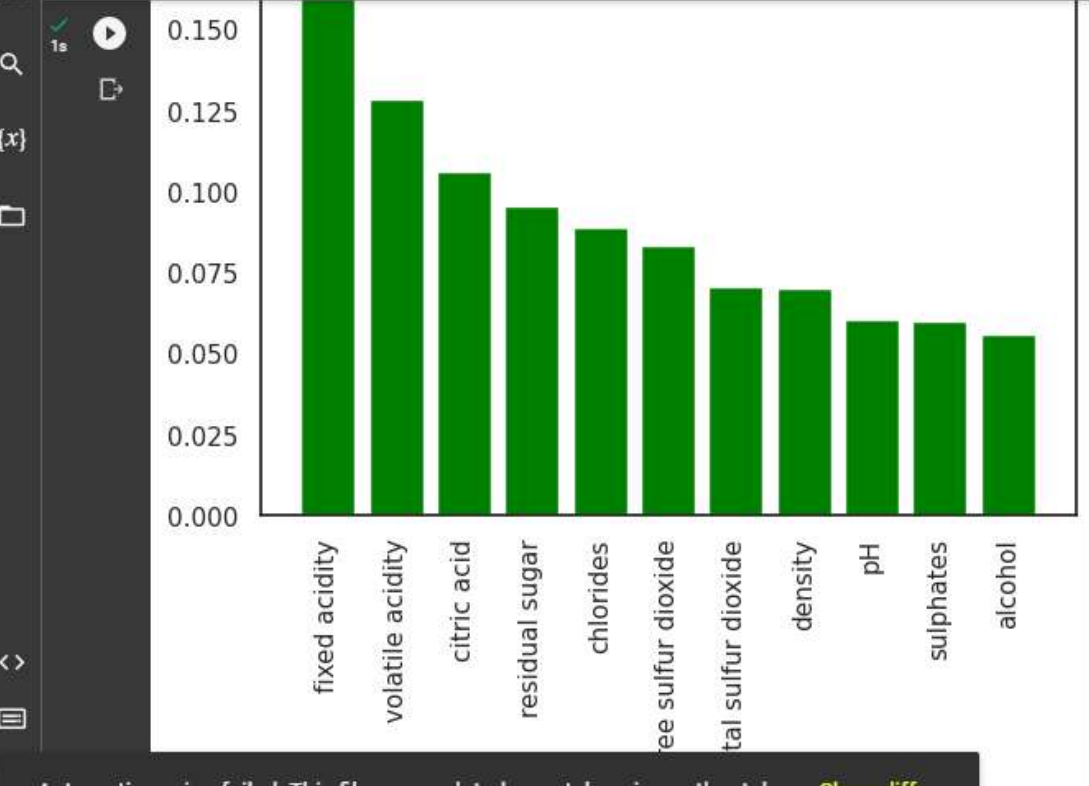
```
[40] for i in range(X.shape[1]):  
      print ("%2d) %-*s %f" % (i + 1, 30, features_label[i], importances[indices[i]]))
```

1)	fixed acidity	0.178690
2)	volatile acidity	0.128748
3)	citric acid	0.106496
4)	residual sugar	0.095718
5)	chlorides	0.089280
6)	free sulfur dioxide	0.083482
7)	total sulfur dioxide	0.070669
8)	density	0.070104
9)	pH	0.060459
10)	sulphates	0.060048
11)	alcohol	0.056307

```
plt.title('Feature Importances')  
plt.bar(range(X.shape[1]), importances[indices], color="green", align="center")  
plt.xticks(range(X.shape[1]), features_label, rotation=90)  
plt.xlim([-1, X.shape[1]])  
plt.show()
```



Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)



Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
+ Code + Text
```

```
[43] from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 5)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train2 = pd.DataFrame(sc.fit_transform(X_train))
X_test2 = pd.DataFrame(sc.transform(X_test))
X_train2.columns = X_train.columns.values
X_test2.columns = X_test.columns.values
X_train2.index = X_train.index.values
X_test2.index = X_test.index.values
X_train = X_train2
X_test = X_test2
```

```
[45] from sklearn.decomposition import PCA
      pca = PCA(n_components = 4)
      X_train = pca.fit_transform(X_train)
      X_test = pca.transform(X_test)
      explained_variance = pca.explained_variance_ratio_
      print(pd.DataFrame(explained_variance))
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
[45] print(pd.DataFrame(explained_variance))
```

```
0 0.281687
1 0.171462
2 0.143245
3 0.114765
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, penalty = 'l2')
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score, recall_score
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
results = pd.DataFrame(['Logistic Regression', acc, prec, rec, f1],
                        columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])
print(results)
```

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.86875	0.6	0.26087	0.363636

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

+ Code + Text RAM Disk

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.86875	0.6	0.26087	0.363636

```
from sklearn.svm import SVC
classifier = SVC(random_state = 0, kernel = 'rbf')
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

model_results = pd.DataFrame([['SVM (RBF)', acc, prec, rec, f1]],
                              columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])

results = results.append(model_results, ignore_index = True)
print(results)
```

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.86875	0.600000	0.260870	0.363636
1	SVM (RBF)	0.87500	0.714286	0.217391	0.333333
2	SVM (RBF)	0.87500	0.714286	0.217391	0.333333

Automatic saving failed. This file was updated remotely or in another tab. Show diff

Classifier = RandomizedSVC(random_state = 0, n_estimators = 100,


```
[51] from sklearn.ensemble import RandomForestClassifier
      classifier = RandomForestClassifier(random_state = 0, n_estimators = 100,
                                         criterion = 'entropy')

      classifier.fit(X_train, y_train)

      # Predicting Test Set
      y_pred = classifier.predict(X_test)
      acc = accuracy_score(y_test, y_pred)
      prec = precision_score(y_test, y_pred)
      rec = recall_score(y_test, y_pred)
      f1 = f1_score(y_test, y_pred)

      model_results = pd.DataFrame([[ 'Random Forest (n=100)', acc, prec, rec, f1]],
                                   columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])

      results = results.append(model_results, ignore_index = True)
      print(results)
```

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.868750	0.600000	0.260870	0.363636
1	SVM (RBF)	0.875000	0.714286	0.217391	0.333333
2	SVM (RBF)	0.875000	0.714286	0.217391	0.333333
3	Random Forest (n=100)	0.903125	0.741935	0.500000	0.597403

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
+ Code + Text
```

```
[52] from sklearn.model_selection import cross_val_score
      accuracies = cross_val_score(estimator = classifier, X= X_train, y = y_train,
                                   cv = 10)
      print("Random Forest Classifier Accuracy: %0.2f (+/- %0.2f)" % (accuracies.mean(), accuracies.std() * 2))

Random Forest Classifier Accuracy: 0.89 (+/- 0.04)
```

```
parameters = {"max_depth": [3, None],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 5, 10],
              "bootstrap": [True, False],
              "criterion": ["entropy"]}

from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator = classifier,
                           param_grid = parameters,
                           scoring = "accuracy",
                           cv = 10,
                           n_jobs = -1)

t0 = time.time()
grid_search = grid_search.fit(X_train, y_train)
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator = classifier,
                           param_grid = parameters,
                           scoring = "accuracy",
                           cv = 10,
                           n_jobs = -1)

t0 = time.time()
grid_search = grid_search.fit(X_train, y_train)
t1 = time.time()
print("Took %.2f seconds" % (t1 - t0))

rf_best_accuracy = grid_search.best_score_
rf_best_parameters = grid_search.best_params_
rf_best_accuracy, rf_best_parameters
```

```
Took 100.39 seconds
(0.8991510826771654,
 {'bootstrap': False,
  'criterion': 'entropy',
  'max_depth': None,
  'min_samples_leaf': 1,
  'min_samples_split': 2})
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
parameters = {"max_depth": [None],  
              'min_samples_split': [8, 10, 12],  
              'min_samples_leaf': [1, 2, 3],  
              "bootstrap": [True],  
              "criterion": ["entropy"]}  
  
from sklearn.model_selection import GridSearchCV  
grid_search = GridSearchCV(estimator = classifier,  
                           param_grid = parameters,  
                           scoring = "accuracy",  
                           cv = 10,  
                           n_jobs = -1)  
  
t0 = time.time()  
grid_search = grid_search.fit(X_train, y_train)  
t1 = time.time()  
print("Took %.2f seconds" % (t1 - t0))  
  
rf_best_accuracy = grid_search.best_score_  
rf_best_parameters = grid_search.best_params_  
rf_best_accuracy, rf_best_parameters
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
{'bootstrap': True,
```



```
[56] Took 30.56 seconds
(0.8881951279527559,
 {'bootstrap': True,
  'criterion': 'entropy',
  'max_depth': None,
  'min_samples_leaf': 1,
  'min_samples_split': 8})
```

```
[57] parameters = {"max_depth": [3, None],
                  'min_samples_split': [2, 5, 10],
                  'min_samples_leaf': [1, 5, 10],
                  "bootstrap": [True, False],
                  "criterion": ["gini"]}
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator = classifier,
                           param_grid = parameters,
                           scoring = "accuracy",
                           cv = 10,
                           n_jobs = -1)

t0 = time.time()
grid_search = grid_search.fit(X_train, y_train)
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
if best_accuracy == grid_search.best_score
```

```
param_grid = parameters,
scoring = "accuracy",
cv = 10,
n_jobs = -1)

t0 = time.time()
grid_search = grid_search.fit(X_train, y_train)
t1 = time.time()
print("Took %.2f seconds" % (t1 - t0))

rf_best_accuracy = grid_search.best_score_
rf_best_parameters = grid_search.best_params_
rf_best_accuracy, rf_best_parameters
```

```
Took 87.44 seconds
(0.8983698326771654,
{'bootstrap': True,
 'criterion': 'gini',
 'max_depth': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2})
```

```
[58] parameters = {"max_depth": [None],
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
+ Code + Text
```

```
[58] parameters = {"max_depth": [None],  
              'min_samples_split': [2, 3, 4],  
              'min_samples_leaf': [8, 10, 12],  
              "bootstrap": [True],  
              "criterion": ["gini"]}  
  
from sklearn.model_selection import GridSearchCV  
grid_search = GridSearchCV(estimator = classifier,  
                           param_grid = parameters,  
                           scoring = "accuracy",  
                           cv = 10,  
                           n_jobs = -1)  
  
t0 = time.time()  
grid_search = grid_search.fit(X_train, y_train)  
t1 = time.time()  
print("Took %.2f seconds" % (t1 - t0))  
  
rf_best_accuracy = grid_search.best_score_  
rf_best_parameters = grid_search.best_params_  
rf_best_accuracy, rf_best_parameters
```

Took 23.19 seconds
(0.8810290761770528)

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

```
[58] rf_best_accuracy, rf_best_parameters

Took 23.19 seconds
(0.8819389763779528,
 {'bootstrap': True,
  'criterion': 'gini',
  'max_depth': None,
  'min_samples_leaf': 8,
  'min_samples_split': 2})
```

```
rf_best_accuracy, rf_best_parameters

(0.8819389763779528,
 {'bootstrap': True,
  'criterion': 'gini',
  'max_depth': None,
  'min_samples_leaf': 8,
  'min_samples_split': 2})
```

```
[60] y_pred = grid_search.predict(X_test)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)


```
y_pred = grid_search.predict(X_test)
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

model_results = pd.DataFrame([[ 'Random Forest (n=100, GSx2 + Gini)', acc, prec, rec, f1]],
                             columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])

results = results.append(model_results, ignore_index = True)
results
```

	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.868750	0.600000	0.260870	0.363636
1	SVM (RBF)	0.875000	0.714286	0.217391	0.333333
2	SVM (RBF)	0.875000	0.714286	0.217391	0.333333
3	Random Forest (n=100)	0.903125	0.741935	0.500000	0.597403
4	Random Forest (n=100, GSx2 + Gini)	0.881250	0.700000	0.304348	0.424242

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)