```
# Sample Machine Learning Project
# Dataset: wine_new.csv


# steps
# 1. data import
# 2. Data Cleaning - Missing values
# 3. Exploratory Data Analysis
# 4. Outlier Detection and Removal
# 5. Data Balancing / Resampling
# 6. Feature Scaling
# 7. Feature selection / Dimentionality Reduction
# 8. Cross Validation
# 9. Algorithm, Hyperparameter Tuning
# 10. Model Building
# 11. Predictions, Performance Monitoring
# 12. Deployment


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



# 1. data import
df = pd.read_csv('wine-new.csv')


df.shape
```

```
(178, 14)
```

```
# 2. Data Cleaning - Missing values
df.isnull().sum()
```

```
class                           0
```

```
alcohol                         0
 malic_acid                     0
 ash                            0
 alcalinity_of_ash              0
 magnesium                      3
 total_phenols                  0
 flavanoids                     0
 nonflavanoid_phenols           0
 proanthocyanins                0
 color_intensity                0
 hue                            2
 od280/od315_of_diluted_wines   7
 proline                        0
dtype: int64
```

```
# check the data form, if string values then encoding you have to do
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   class                         178 non-null    int64
 1   alcohol                       178 non-null    float64
 2    malic_acid                   178 non-null    float64
 3    ash                          178 non-null    float64
 4    alcalinity_of_ash            178 non-null    float64
 5    magnesium                    175 non-null    float64
 6    total_phenols                178 non-null    float64
 7    flavanoids                   178 non-null    float64
 8    nonflavanoid_phenols         178 non-null    float64
 9    proanthocyanins              178 non-null    float64
 10   color_intensity              178 non-null    float64
 11   hue                          176 non-null    float64
 12   od280/od315_of_diluted_wines 171 non-null    float64
 13   proline                      178 non-null    int64
dtypes: float64(12), int64(2)
memory usage: 19.6 KB
```

```
df.columns = df.columns.str.strip()
# space is thter infrom of clumnname
```

```
df.columns
```

```
Index(['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash',
       'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
       'proanthocyanins', 'color_intensity', 'hue',
       'od280/od315_of_diluted_wines', 'proline'],
      dtype='object')
```

```
# 2. Data Cleaning
df.fillna(method='pad', inplace=True)
```

```
df.isnull().sum()
```

```
class                           0
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
dtype: int64
```

```
# seperate the input output data
x = df.drop('class', axis = 1)
y = df['class']
```

```
x.shape
```

```
(178, 13)
```

```
# 3. Exploratory Data Analysis
x.columns
```

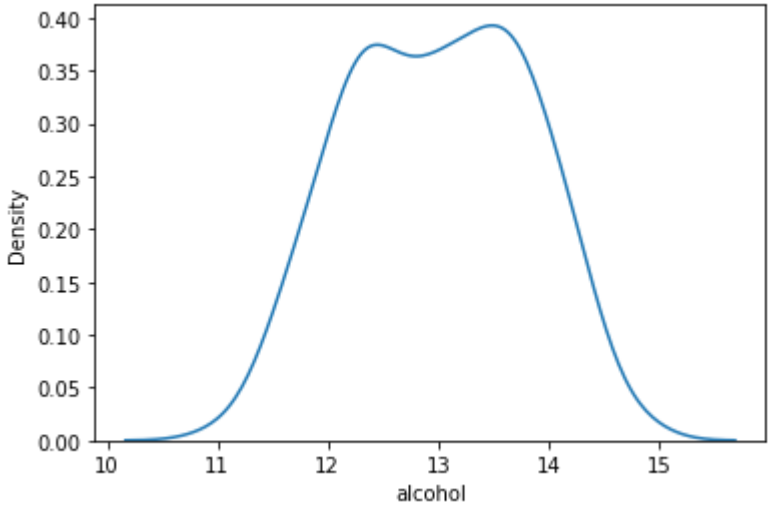
```
Index(['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
       'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
       'proanthocyanins', 'color_intensity', 'hue',
       'od280/od315_of_diluted_wines', 'proline'],
      dtype='object')
```

```
x.describe()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenol
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.00000
mean	13.000618	2.336348	2.366517	19.494944	99.511236	2.295112	2.029270	0.36185
std	0.811827	1.117146	0.274344	3.339564	14.125842	0.625851	0.998859	0.12445
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.13000
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.27000
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.34000
75%	13.677500	3.082500	2.557500	21.500000	106.750000	2.800000	2.875000	0.43750
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.66000

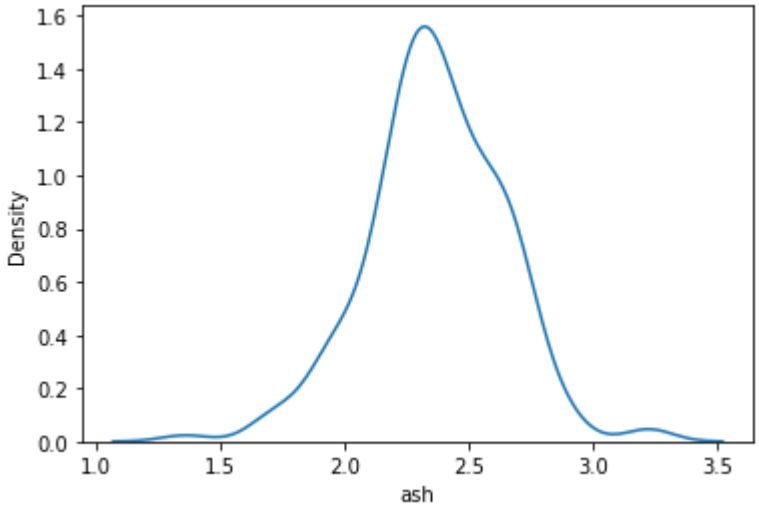
```
sns.kdeplot(df['alcohol'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0748512690>
```




```
sns.kdeplot(df['ash'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0746435750>
```



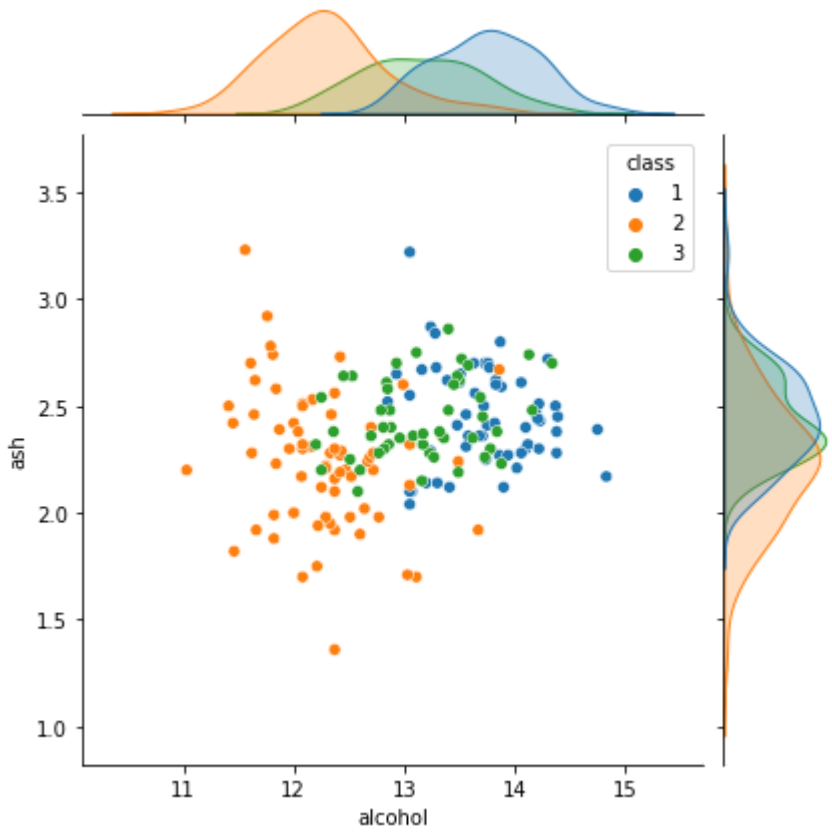

```
# with the help of pairplot we can check for the clusters
# check for grouping
```

```
sns.pairplot(df, hue='class',palette='tab10')
```

```
<seaborn.axisgrid.PairGrid at 0x7f0745f6ec50>
```


```
sns.jointplot(data=df, x = 'alcohol', y='ash',
              hue='class', palette='tab10')
```

```
<seaborn.axisgrid.JointGrid at 0x7f07409c93d0>
```




```
sns.countplot(x = y)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f07409b09d0>
```


```
y.value_counts()
```

```
2    71
1    59
3    48
Name: class, dtype: int64
```

```
# 4. Outlier Detection and Removal
from sklearn.ensemble import IsolationForest
iso = IsolationForest(random_state=0, contamination=0.05)
clean = iso.fit_predict(x, y)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but Isola
  "X does not have valid feature names, but"
```

```
x = x[clean == 1]
y = y[clean == 1]
```

```
x.shape
```

```
(169, 13)
```

```
# 5. Data Balancing / Resampling
y.value_counts()
```

```
2    66
1    57
3    46
Name: class, dtype: int64
```

```
# random sampling match with original data
```

```
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(random_state=0)
x_res, y_res = ros.fit_resample(x, y)
```

```
x_res.shape
```

```
(198, 13)
```

```
y_res.value_counts()
```

```
1    66
2    66
3    66
Name: class, dtype: int64
```

```
# 6. Feature Scaling
# it required when, all features are not in similar range
```

```
x_res.describe()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenol
count	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000	198.00000

```
x_new=x_res.drop('proline' , axis=1)
plt.figure(figsize=(16,9))
plt.boxplot(x_new, labels=x_new.columns);
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-56-f63b97c6666b> in <module>()
      1 x_new=x_res.drop('proline' , axis=1)
      2 plt.figure(figsize=(16,9))
----> 3 plt.boxplot(x_new, labels=x_new.columns);

autorange)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x_res)
```

```
    1173     input_whis = whis
```

```
x_scaled = pd.DataFrame(x_scaled, columns=x_res.columns)
```

```
x_scaled.describe()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenol
count	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000	198.000000	198.00000
mean	0.528057	0.342137	0.449595	0.452181	0.422266	0.410653	0.421484	0.45292
std	0.200547	0.230430	0.164130	0.165589	0.182239	0.219753	0.289037	0.23875
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	0.365789	0.177866	0.356908	0.336436	0.275362	0.215827	0.112717	0.26415
50%	0.538158	0.245059	0.434211	0.444149	0.405797	0.395683	0.450867	0.39622
75%	0.681579	0.497036	0.565789	0.547872	0.521739	0.607014	0.667630	0.64150
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

```
# 7. Feature selection / Dimentionality Reduction
```

```
from sklearn.feature_selection import SelectKBest, chi2
```

```
skf = SelectKBest(score_func=chi2, k=5)
# we are selecting top 5 features
```

```
x_new = skf.fit_transform(x_scaled, y_res)
```

```
x_new.shape
```

```
(198, 5)
```

```
skf.get_support()
# true means feature are selected
```

```
array([False, False, False, False, False,  True,  True, False, False,
       True, False,  True,  True])
```

```
x_new = x_scaled.iloc[:,skf.get_support()]
```

```
x_new
# based on skf top 5 features are selected
```

	total_phenols	flavanoids	color_intensity	od280/od315_of_diluted_wines	proline
0	0.611511	0.748555	0.416428	0.970696	0.561341
1	0.557554	0.661850	0.296084	0.780220	0.550642
2	0.611511	0.800578	0.420248	0.695971	0.646933
3	0.611511	0.641618	0.290353	0.608059	0.325963
4	0.780576	0.843931	0.522445	0.578755	0.835949
...	...	...	...	...	...
193	0.323741	0.095376	0.297994	0.285714	0.194009
194	0.187050	0.002890	0.422159	0.201465	0.215407
195	0.438849	0.037572	0.347660	0.322344	0.222539

```
# 8. cross validation
from sklearn.model_selection import train_test_split
```

```
x_train,·x_test,·y_train,·y_test·=·train_test_split(
····x_new,·y_res,·random_state=0,·stratify=y_res
)
```

```
x_train.shape
```

```
(148, 5)
```

```
x_test.shape
```

```
(50, 5)
```

```
y_train.value_counts()
```

```
3    50
2    49
```

```
1    49
Name: class, dtype: int64
```

```
# 9. Algorithm, Hyperparameter Tuning

# we can try diffirent different algorithm for more accuracy


from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import plot_confusion_matrix


from sklearn.model_selection import GridSearchCV


params = {
    'random_state': [0,1,2,3,4,5],
    'n_estimators': [10,20,30,40,50,100],
    'criterion': ['gini','entropy']
}
# # identify the best paramenter for random forest


rf = RandomForestClassifier()
grid = GridSearchCV(rf, param_grid=params, cv=5,
                    scoring='accuracy')


grid.fit(x_train,·y_train)
y_pred·=·grid.predict(x_test)


accuracy_score(y_test, y_pred)
```

```
0.94
```

```
grid.best_estimator_
```

```
RandomForestClassifier(n_estimators=10, random_state=4)
```

```
#select the best feature for that algo
```

```
from sklearn.feature_selection import RFE
```

```
rfe = RFE(rf, n_features_to_select=5)
```

```
rfe.fit(x_scaled, y_res)
```

```
RFE(estimator=RandomForestClassifier(), n_features_to_select=5)
```

```
rfe.get_support()
```

```
array([False, False, False, False, False, False,  True, False, False,
       True,  True,  True,  True])
```

```
x_new = x_scaled.iloc[:,rfe.get_support()]
```

```
x_train, x_test, y_train, y_test = train_test_split(
    x_new, y_res, random_state=0, stratify=y_res
)
```

```
rf.fit(x_train,y_train)
y_pred = rf.predict(x_test)
```

```
accuracy_score(y_test, y_pred)
# we change the features thats why accuracy change
```

```
0.96
```

```
# using grid we will check accuracy
```

```
# using grid same accuracy we are gettting
# hence model is ready


rf = RandomForestClassifier()
grid = GridSearchCV(rf, param_grid=params, cv=5,
                    scoring='accuracy')


grid.fit(x_train, y_train)
y_pred = grid.predict(x_test)
accuracy_score(y_test, y_pred)
```

```
0.96
```

```
# Serialization


import joblib


joblib.dump(grid, 'classifier.model')
# this model save in current directory
```

```
['classifier.model']
```

```
x_new.columns
```

```
Index(['flavanoids', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines',
       'proline'],
      dtype='object')
```

✓ 0s completed at 3:34 PM