Вступ до машинного навчання

Джерела: \ https://github.com/amueller/introduction_to_ml_with_python (https://github.com/amueller/introduction_to_ml_with_python)

Демостраційна частина

In [1]:
```
# перший виклик matplotlib займає певний час, це нормально
%matplotlib inline
```

Essential Libraries and Tools

NumPy

In [2]:
```
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))
```

```
x:
[[1 2 3]
 [4 5 6]]
```

SciPy

In [3]:
```
from scipy import sparse

# Create a 2D NumPy array with a diagonal of ones, and zeros everywhere else
eye = np.eye(4)
print("NumPy array:\n", eye)
```

```
NumPy array:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

In [4]:
```
# Convert the NumPy array to a SciPy sparse matrix in CSR format
# Only the nonzero entries are stored
sparse_matrix = sparse.csr_matrix(eye)
print("\nSciPy sparse CSR matrix:\n", sparse_matrix)
```

```
SciPy sparse CSR matrix:
   (0, 0)       1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
```

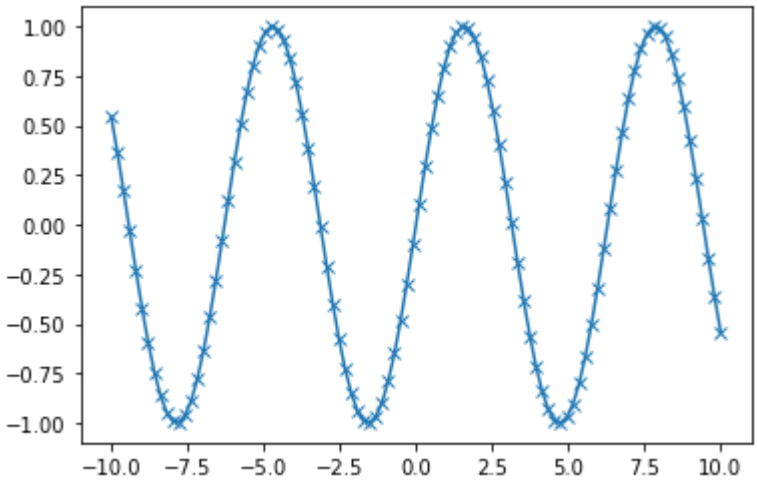
In [5]:
```
data = np.ones(4)
row_indices = np.arange(4)
col_indices = np.arange(4)
eye_coo = sparse.coo_matrix((data, (row_indices, col_indices)))
print("COO representation:\n", eye_coo)
```

```
COO representation:
   (0, 0)       1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
```

matplotlib

In [6]:
```
%matplotlib inline
import matplotlib.pyplot as plt

# Generate a sequence of numbers from -10 to 10 with 100 steps in between
x = np.linspace(-10, 10, 100)
# Create a second array using sine
y = np.sin(x)
# The plot function makes a line chart of one array against another
plt.plot(x, y, marker="x")
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x1dec72dc730>]
```



pandas

In [7]:
```
import pandas as pd

# create a simple dataset of people
data = {'Name': ["John", "Anna", "Peter", "Linda"],
        'Location' : ["New York", "Paris", "Berlin", "London"],
        'Age' : [24, 13, 53, 33]
       }

data_pandas = pd.DataFrame(data)
# IPython.display allows "pretty printing" of dataframes
# in the Jupyter notebook
display(data_pandas)
```

	Name	Location	Age
0	John	New York	24
1	Anna	Paris	13
2	Peter	Berlin	53
3	Linda	London	33

In [8]:
```
# Select all rows that have an age column greater than 30
display(data_pandas[data_pandas.Age > 30])
```

	Name	Location	Age
2	Peter	Berlin	53
3	Linda	London	33

Version check

In [9]:
```
import sys
print("Python version:", sys.version)

import pandas as pd
print("pandas version:", pd.__version__)

import matplotlib
print("matplotlib version:", matplotlib.__version__)

import numpy as np
print("NumPy version:", np.__version__)

import scipy as sp
print("SciPy version:", sp.__version__)

import IPython
print("IPython version:", IPython.__version__)

import sklearn
print("scikit-learn version:", sklearn.__version__)
```

```
Python version: 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.192
4 64 bit (AMD64)]
pandas version: 1.1.2
matplotlib version: 3.3.2
NumPy version: 1.19.2
SciPy version: 1.5.2
IPython version: 7.18.1
scikit-learn version: 0.23.2
```

A First Application: Classifying Iris Species

Meet the Data

In [10]:
```
from sklearn.datasets import load_iris
iris_dataset = load_iris()
```

In [11]:
```
print("Keys of iris_dataset:\n", iris_dataset.keys())
```

```
Keys of iris_dataset:
 dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_name
s', 'filename'])
```

In [12]:
```
print(iris_dataset['DESCR'][:193] + "\n...")
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
```

In [13]:
```
print("Target names:", iris_dataset['target_names'])
```

```
Target names: ['setosa' 'versicolor' 'virginica']
```

In [14]:
```
print("Feature names:\n", iris_dataset['feature_names'])
```

```
Feature names:
 ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

In [15]:
```
print("Type of data:", type(iris_dataset['data']))
```

```
Type of data: <class 'numpy.ndarray'>
```

In [16]:
```
print("Shape of data:", iris_dataset['data'].shape)
```

```
Shape of data: (150, 4)
```

In [17]:
```
print("First five rows of data:\n", iris_dataset['data'][:5])
```

```
First five rows of data:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

In [18]:
```
print("Type of target:", type(iris_dataset['target']))
```

```
Type of target: <class 'numpy.ndarray'>
```

In [19]:
```
print("Shape of target:", iris_dataset['target'].shape)
```

```
Shape of target: (150,)
```

In [20]:
```
print("Target:\n", iris_dataset['target'])
```

```
Target:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Measuring Success: Training and Testing Data

In [21]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)
```

In [22]:
```
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
```

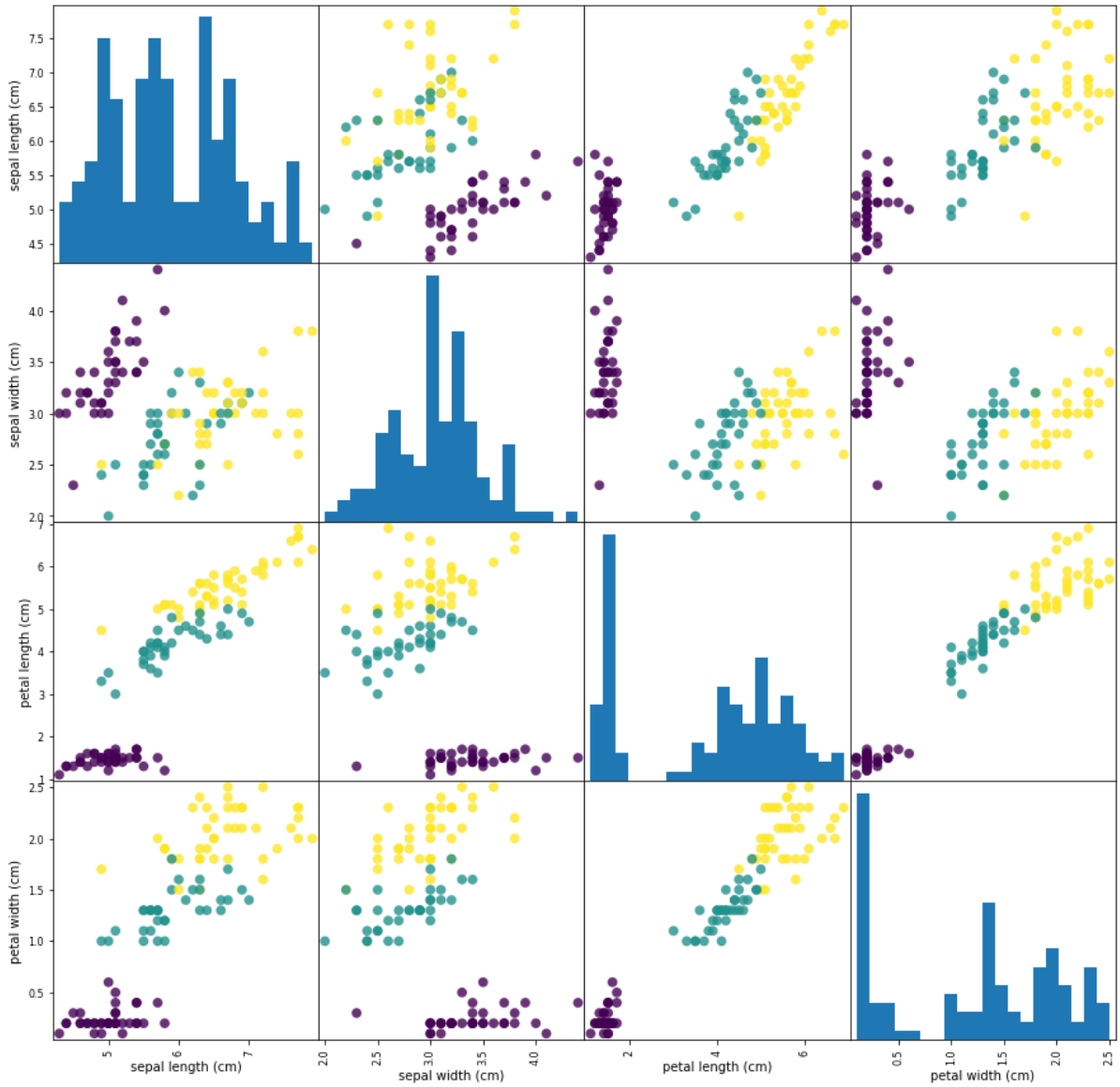
```
X_train shape: (112, 4)
y_train shape: (112,)
```

In [23]:
```
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_test shape: (38, 4)
y_test shape: (38,)
```

First Things First: Look at Your Data

In [24]:
```
# create dataframe from data in X_train
# label the columns using the strings in iris_dataset.feature_names
iris_dataframe = pd.DataFrame(X_train, columns=iris_dataset.feature_names)
# create a scatter matrix from the dataframe, color by y_train
pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15),
                           marker='o', hist_kwds={'bins': 20}, s=60,
                           alpha=.8);
```



Building Your First Model: k-Nearest Neighbors

In [25]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

In [26]:
```
knn.fit(X_train, y_train)
```

Out[26]:
```
KNeighborsClassifier(n_neighbors=1)
```

Making Predictions

In [27]:
```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape:", X_new.shape)
```

```
X_new.shape: (1, 4)
```

In [28]:
```
prediction = knn.predict(X_new)
print("Prediction:", prediction)
print("Predicted target name:",
       iris_dataset['target_names'][prediction])
```

```
Prediction: [0]
Predicted target name: ['setosa']
```

Evaluating the Model

In [29]:
```
y_pred = knn.predict(X_test)
print("Test set predictions:\n", y_pred)
```

```
Test set predictions:
 [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2]
```

In [30]:
```
print("Test set score: {:.2f}".format(np.mean(y_pred == y_test)))
```

```
Test set score: 0.97
```

In [31]:
```
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
Test set score: 0.97
```

Summary and Outlook

In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)

print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
Test set score: 0.97
```

Завдання

Частина І

1. Яку за типом задачу машинного навчання було розглянуто вище?

Відповідь: задача багатоклассової (3-классової класифікації)

1. Назвіть трійку <E,T,P>, що їй відповідає

Відповідь:

- E - датасет ознак квіток та відповідних типів
- T - передбачити тип квітки (1 з 3х)
- P - точність (відсоток правильно передбачених квіток)

1. На основі заданої вибірки, могли б Ви запропонувати іншу задачу машинного навчання? Яку?

Можливі задачі:

- Кластеризація
- Регресія відносно якоїсь з ознак (наприклад, передбачити sepal length для нової квітки з відомими іншими змінними)

1. Прокоментуйте графіки (див. вище), що ілюструють ко-залежність ознак між собою

Залежність ознак між собою демонструє гарну лінійну сепарабельність точок в залежності від класів, деякі ознаки мають високу кореляцію (особливо petal_width - petal_length або petal_length - sepal_length), що дозволить простому класифікатору (або SVC) гарно виконати поставлену задачу

1. Наведіть приклади трьох задач навчання (заданих трійками <E,T,P>), відмінні від наведених на лекції

- Time-series Price Forecasting - Передбачення ціни товару через проміжок часу : {E : історичні дані ціни товару за останні роки , T : передбачена ціна , P : різні (наприклад, MAPE)}
- Sentiment Analysis - Визначити інтонацію тексту-звернення до якоїсь організації : {E : принаймні частково-розмічені тексти попередніх звернень , T : визначити, чи є текст негативним, нейтральним, або позитивним , P : F1-score}
- NER : {E : частково-розмічені дані (розмічення таких даних є однією з основних проблем NLP), T : знайти в тексті "об'єктні сутності" (імена людей,місця, дати подій) , P : точність (кількість знайдених об'єктів в конкретному фрагменті) }

Частина II

Повторіть приготування набору даних до тренування моделі для вибірки про класифікацію вина. Не зневажайте переглядом описового файлу. Напишіть власний короткий опис до датасету, використовуючи знання, отримані при дослідженні вибірки (щонайменше 7 речень).

In [33]:
```
#data preparation
from sklearn.datasets import load_wine
wine = load_wine()

print(wine['DESCR'][:1772])
```

.. _wine_dataset:

Wine recognition dataset

Data Set Characteristics:

    :Number of Instances: 178 (50 in each of three classes)
    :Number of Attributes: 13 numeric, predictive attributes and the class
    :Attribute Information:
                - Alcohol
                - Malic acid
                - Ash
                - Alcalinity of ash
                - Magnesium
                - Total phenols
                - Flavanoids
                - Nonflavanoid phenols
                - Proanthocyanins
                - Color intensity
                - Hue
                - OD280/OD315 of diluted wines
                - Proline

    - class:
            - class_0
            - class_1
            - class_2

    :Summary Statistics:

    ============================== ==== ===== ======= =====
                                    Min   Max   Mean   SD
    ============================== ==== ===== ======= =====
    Alcohol: 11.0 14.8   13.0   0.8
    Malic Acid: 0.74 5.80   2.34 1.12
    Ash: 1.36 3.23   2.36 0.27
    Alcalinity of Ash: 10.6 30.0   19.5   3.3
    Magnesium: 70.0 162.0   99.7 14.3
    Total Phenols: 0.98 3.88   2.29 0.63
    Flavanoids: 0.34 5.08   2.03 1.00
    Nonflavanoid Phenols: 0.13 0.66   0.36 0.12
    Proanthocyanins: 0.41 3.58   1.59 0.57
    Colour Intensity: 1.3 13.0   5.1   2.3
    Hue: 0.48 1.71   0.96 0.23
    OD280/OD315 of diluted wines: 1.27 4.00   2.61 0.71
    Proline: 278 1680   746   315
    ============================== ==== ===== ======= =====

    :Missing Attribute Values: None
    :Class Distribution: class_0 (59), class_1 (71), class_2 (48)
    :Creator: R.A. Fisher
    :Donor: Michael Marshall

In [34]:
```
wine.keys()
```

Out[34]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_name
s'])
```

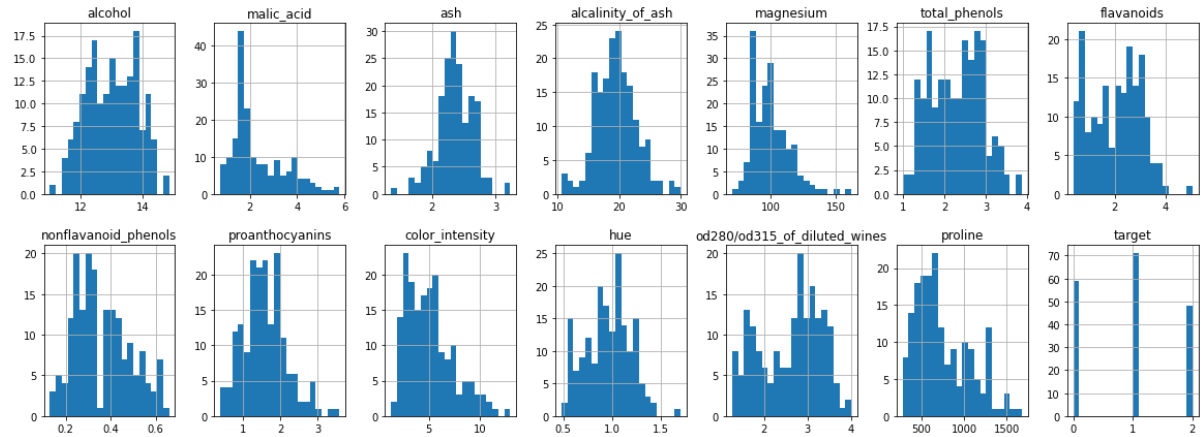
In [35]:
```
wine_df = pd.DataFrame(data=np.c_[wine['data'],wine['target']],columns = wine[
'feature_names']+['target'])

target_names_map = dict(zip([*range(len(wine['target_names']))],wine['target_n
ames']))
# wine_df['target'] = wine_df['target'].map(target_names_map) # if we want to
 map target values with names of its classes

wine_df.head()
```

Out[35]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavano
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

Feature exploration

In [36]:
```
#feature distribution
wine_df.hist(bins=20, figsize=(20,7),layout=(2,7));
```



In [37]:
```
#correlation matrix
wine_df.corr()
```

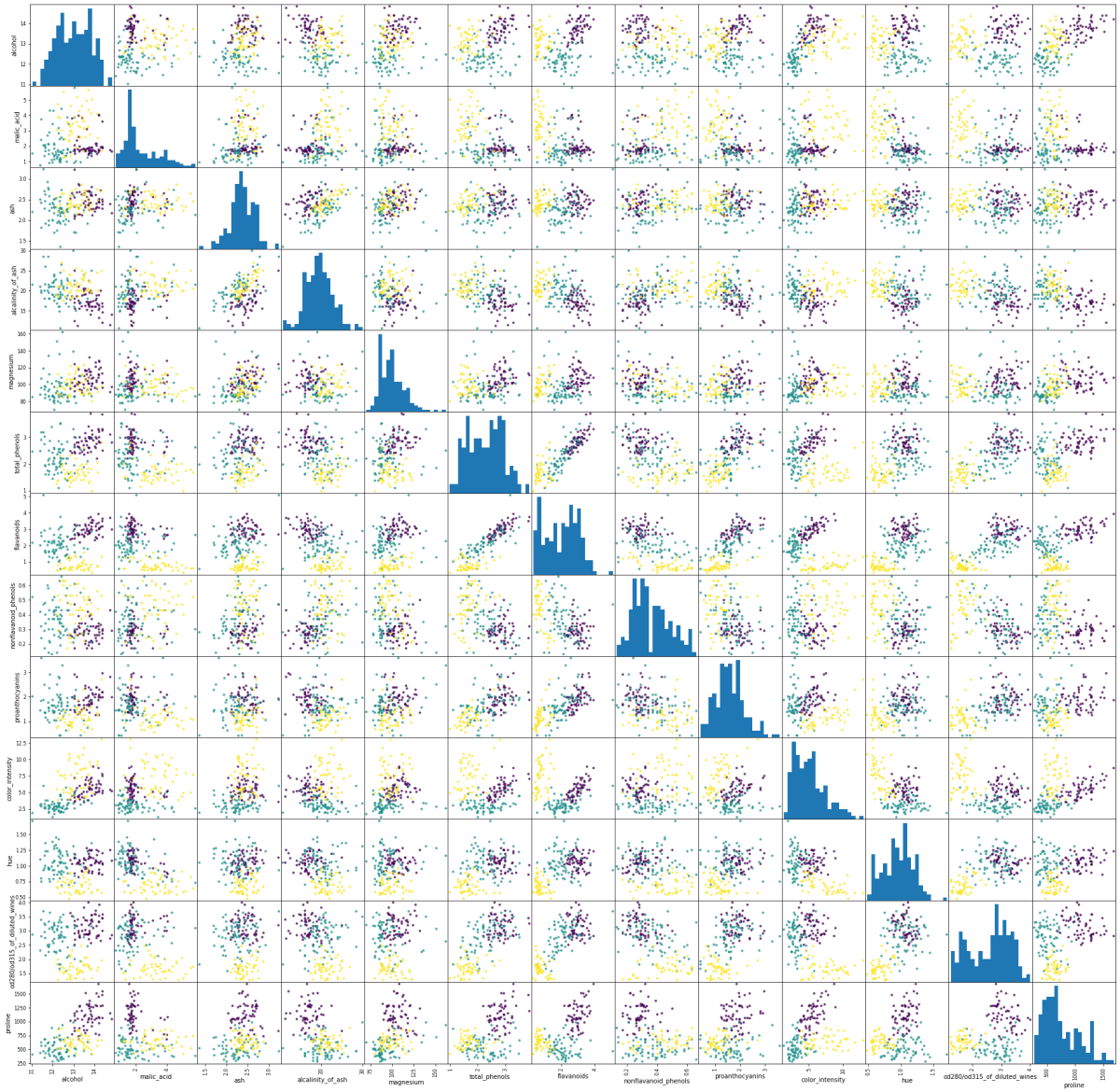
Out[37]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	tot
alcohol	1.000000	0.094397	0.211545	-0.310235	0.270798	
malic_acid	0.094397	1.000000	0.164045	0.288500	-0.054575	
ash	0.211545	0.164045	1.000000	0.443367	0.286587	
alcalinity_of_ash	-0.310235	0.288500	0.443367	1.000000	-0.083333	
magnesium	0.270798	-0.054575	0.286587	-0.083333	1.000000	
total_phenols	0.289101	-0.335167	0.128980	-0.321113	0.214401	
flavanoids	0.236815	-0.411007	0.115077	-0.351370	0.195784	
nonflavanoid_phenols	-0.155929	0.292977	0.186230	0.361922	-0.256294	
proanthocyanins	0.136698	-0.220746	0.009652	-0.197327	0.236441	
color_intensity	0.546364	0.248985	0.258887	0.018732	0.199950	
hue	-0.071747	-0.561296	-0.074667	-0.273955	0.055398	
od280/od315_of_diluted_wines	0.072343	-0.368710	0.003911	-0.276769	0.066004	
proline	0.643720	-0.192011	0.223626	-0.440597	0.393351	
target	-0.328222	0.437776	-0.049643	0.517859	-0.209179	

In [38]:
```
#correlation between target and other features
wine_df.corr()['target'].sort_values(ascending=False)
```

Out[38]:
```
target                          1.000000
alcalinity_of_ash               0.517859
nonflavanoid_phenols            0.489109
malic_acid                      0.437776
color_intensity                 0.265668
ash                            -0.049643
magnesium                      -0.209179
alcohol                        -0.328222
proanthocyanins                -0.499130
hue                            -0.617369
proline                        -0.633717
total_phenols                  -0.719163
od280/od315_of_diluted_wines   -0.788230
flavanoids                     -0.847498
Name: target, dtype: float64
```

In [39]:
```
#higher resolution graphs
# matplotlib.rcParams["figure.dpi"] = 200

attributes = ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash',
              'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenol
s', 'proanthocyanins',
 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline']
pd.plotting.scatter_matrix(wine_df[attributes],
                c=wine_df['target'],
                alpha=0.8,# marker='o',
                s=60,
                hist_kwds={'bins':20},
                figsize=(32,32))
plt.show()
```



Короткий опис датасету

- Wine dataset consists of 178 samples of 3 slightly differently distributed sorts of wine. (Uneven distribution may cause problems!)
- Each sample consists of 13 numerical features, each describing some property of wine (i.e., degree of alcohol).
- Among the features,2 (magnesium and proline) are integers, and other 11 are floats, all of them being continuous and positive (and may be normalized) .
- Target variable has high negative correlation with a level of flavanoids and od280/od315 and almost zero correlation with ash levels.
- Most other features have small correlations with target variable and generally don't highly correlate with each other (with an exception of high correlation between flavanoids and total_phenols).
- Also, it doesn't seem there are easily-traceable non-linear relationships between features.
- In comparison with Iris dataset, data cannot be linearly separated between classes so easily, meaning that performing a classification task will be a little harder.
- It intuitively seems that classification task be solved well via KNN or Desicion Tree Classifier.

Stratified train_test split

Ми використовуємо stratified split для збереження розподілу класів в тестових даних (реально тут це не дуже важливо, оскільки класи розподілени достатньо рівномірно, але це точно не погіршує результати)

In [40]:
```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits = 1,
                               test_size = 0.2,
                               random_state = 42)
for train_ind,test_ind in split.split(wine_df.drop(columns='target',axis=1), w
ine_df['target']):
    train = wine_df.loc[train_ind]
    test = wine_df.loc[test_ind]

X_train,y_train = train.drop('target',axis=1), train['target']
X_test, y_test = test.drop('target',axis=1), test['target']

X_train.shape, y_train.shape,X_test.shape,y_test.shape
```

Out[40]:
```
((142, 13), (142,), (36, 13), (36,))
```

Primitive modelling

In [41]:
```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier().fit(X_train,y_train)
print(f"train accuracy: {np.mean(y_train==clf.predict(X_train))}\ntest accurac
y = {np.mean(y_test == clf.predict(X_test)):.4f}")
```

```
train accuracy: 0.7816901408450704
test accuracy = 0.8056
```

In [42]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train,y_train)
print(f"train accuracy: {np.mean(y_train==clf.predict(X_train))}\ntest accurac
y = {np.mean(y_test == clf.predict(X_test)):.4f}")
```

```
train accuracy: 1.0
test accuracy = 0.9444
```

In [43]:
```
#comparison with svc
from sklearn.svm import SVC
clf = SVC().fit(X_train,y_train)
print(f"train accuracy: {np.mean(y_train==clf.predict(X_train))}\ntest accurac
y = {np.mean(y_test == clf.predict(X_test)):.4f}")
```

```
train accuracy: 0.676056338028169
test accuracy = 0.6944
```

In [ ]:
```

```