28 Εισαγωγη στη Μηχανικη Μαθηση - SciKit-Learn II

28.0.1 Απαντήσεις ερωτήσεων προηγούμενου μαθήματος

Τι είναι η βιβλιοθήκη scikit-learn;

Η scikit-learn είναι μια ανοικτού κώδικα βιβλιοθήκη για τη μηχανική μάθηση στην Python.

Τι είναι η μηχανική μάθηση;

Η μηχανική μάθηση είναι ένα πεδίο της τεχνητής νοημοσύνης που επιτρέπει στους υπολογιστές να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις ή να προσπαθούν να επιλύσουν προβλήματα χωρίς να προγραμματίζονται εξαντλητικά.

Τι είναι ένας ταξινομητής (classifier);

Ένας ταξινομητής είναι ένα μοντέλο μηχανικής μάθησης που χρησιμοποιείται για να κατατάξει ή να ταξινομήσει δεδομένα σε κατηγορίες ή τάξεις.

Ποια είναι η διαδικασία της σταθεροποίησης των τυχαίων αριθμών στο scikit-learn;

Μπορούμε να χρησιμοποιήσουμε το random_state για να ορίσουμε ένα σταθερό σπόρο (seed) για την αναπαράσταση τυχαίων αριθμών σε επαναληπτικές διαδικασίες.

Τι είναι ο διαχωρισμός των δεδομένων σε σύνολο εκπαίδευσης και σύνολο ελέγχου;

Ο διαχωρισμός των δεδομένων σε σύνολο εκπαίδευσης και σύνολο ελέγχου είναι η διαδικασία κατά την οποία τα δεδομένα χωρίζονται σε δύο σύνολα, ένα που χρησιμοποιείται για την εκπαίδευση του μοντέλου και ένα για τον έλεγχο της απόδοσής του.

Τι είναι ο ταξινομητής RandomForestClassifier;

Ο ταξινομητής RandomForestClassifier είναι ένα μοντέλο μηχανικής μάθησης που βασίζεται σε ένα δάσος από δέντρα αποφάσεων και χρησιμοποιείται για την ταξινόμηση των δεδομένων.

Τι είναι μια υπερ-παράμετρος (hyperparameter);

Οι υπερ-παράμετροι είναι παράμετροι που δεν μαθαίνονται από το μοντέλο αλλά πρέπει να οριστούν πριν την εκπαίδευση και επηρεάζουν την απόδοση του μοντέλου.

Τι είναι η διαδικασία της διασταυρούμενης επικύρωσης (cross-validation);

Η διαδικασία της διασταυρούμενης επικύρωσης είναι μια τεχνική που χρησιμοποιείται για να εκτιμήσει την απόδοση ενός μοντέλου μηχανικής μάθησης, διαχωρίζοντας τα δεδομένα σε πολλά υποσύνολα και εκπαιδεύοντας/ελέγχοντας το μοντέλο πολλές φορές.

Τι είναι το πλέγμα υπερ-παραμέτρων (hyperparameter grid);

Το πλέγμα υπερ-παραμέτρων είναι ένα σύνολο πιθανών τιμών για τις υπερ-παραμέτρους ενός μοντέλου, που χρησιμοποιείται κατά την διασταυρούμενη επικύρωση για την επιλογή των βέλτιστων υπερ-παραμέτρων.

Τι κάνει η συνάρτηση GridSearchCV;

Η συνάρτηση GridSearchCV εκτελεί μια διασταυρούμενη αναζήτηση στο πλέγμα υπερ-παραμέτρων για να βρει τις βέλτιστες υπερ-παραμέτρους για ένα μοντέλο.

Πώς μπορούμε να αποκτήσουμε τις καλύτερες υπερ-παραμέτρους μετά από μια αναζήτηση GridSearchCV;

Μπορούμε να χρησιμοποιήσουμε το αποτέλεσμα της αναζήτησης GridSearchCV με τη μέθοδο .best_params_.

Τι κριτήρια μπορούμε να χρησιμοποιήσουμε για την δημιουργία ενός δέντρου αποφάσεων στο scikit-learn;

Μπορούμε να χρησιμοποιήσουμε τα κριτήρια "gini" ή "entropy" για τη δημιουργία ενός δέντρου αποφάσεων.

Πώς μπορούμε να περιορίσουμε το βάθος ενός δέντρου αποφάσεων στο scikit-learn;

Μπορούμε να χρησιμοποιήσουμε την υπερ-παράμετρα "max_depth" για να περιορίσουμε το βάθος ενός δέντρου.

Τι υπερ-παραμέτρους μπορούμε να ρυθμίσουμε για τον ταξινομητή RandomForestClassifier στο scikit-learn;

Ορισμένες υπερ-παράμετροι που μπορούμε να ρυθμίσουμε για τον ταξινομητή RandomForestClassifier περιλαμβάνουν τον αριθμό των δέντρων ("n_estimators"), το κριτήριο ("criterion"), το μέγιστο βάθος ("max_depth"), και τον μέγιστο αριθμό των χαρακτηριστικών ("max_features").

28.1.0 Εφαρμογή Μοντέλων Machine Learning με το SciKit-Learn

Ολοκληρώνουμε σήμερα την εισαγωγή μας στην Επιστήμη των Δεδομένων και ειδικότερα στη Μηχανική Μάθηση, με ένα project, στο οποίο θα εφαρμόσουμε διάφορα μοντέλα χρησιμοποιώντας τη βιβλιοθήκη Scikit-learn. Η Scikit-Learn είναι η βασική βιβλιοθήκη για τη μηχανική μάθηση. Μας επιτρέπει να διαχειριζόμαστε δεδομένα, να χρησιμοποιούμε μοντέλα AI και πολλά άλλα!

Η Scikit-Learn είναι ιδανική για την επεξεργασία αριθμητικών δεδομένων καθώς και κειμένων. Για να κάνουμε Μηχανική Εκμάθηση σε εικόνες, συμβουλεύουμε μάλλον τις βιβλιοθήκες Deep Learning όπως η Keras, τις οποίες έχουμε φτιάξει ένα σεμινάριο εδώ!

Σε αυτό το μάθημα θα κάνουμε πρώτα μια Εξερευνητική Ανάλυση Δεδομένων (**Exploratory Data Analysis** - EDA) για να κατανοήσουμε το σύνολο δεδομένων μας και να αποφασίσουμε ποια μοντέλα θα χρησιμοποιήσουμε για Μηχανική Μάθηση.

Θα δούμε διαφορετικές τεχνικές EDA και στη συνέχεια θα δούμε 4 μοντέλα Machine Learning:

- Logistic Regression
- Υποστήριξη Μηχανών Διανυσμάτων - Support Vector Machines – SVM
- Στοχαστική Κάθοδος Κλίσης - Stochastic Gradient Descent
- Decision Tree - Δέντρο απόφασης

28.1.2 Wine Dataset - Δεδομένα κρασιού

Χρειαζόμαστε ένα dataset για να πειραματιζόμαστε. Θα χρησιμοποιήσουμε ένα αρκετά διαδεδομένο δημόσιο dataset, το οποίο περιέχει δεδομένα κρασιού και είναι το winequality-white. Το έχουμε ήδη ανεβάσει στο Google Drive και μπορείτε να το κατεβάσετε από τον παραπάνω σύνδεσμο. Είναι ένα αρχείο το οποίο προέρχεται από τον ιστότοπο Kaggle, όπως και προηγούμενο dataset που ήδη χρησιμοποιήσαμε.

Σ' αυτό το αρχείο, υπάρχουν διάφορες μετρήσεις, με σκοπό να διαπιστώνεται η ποιότητα του κρασιού.

Ξεκινάμε με την εισαγωγή της Pandas, την οποία έχουμε ήδη γνωρίσει, ώστε να γνωρίσουμε και τα δεδομένα μας.

```
import pandas as pd
```

Μόλις κατεβάσουμε το αρχείο, το τοποθετούμε σε ένα φάκελο «βολικό» για τη χρήση μας. Προτείνουμε προσωρινά, να το έχετε στο βασικό φάκελο της Python. Στη συνέχεια, μπορούμε να το φορτώσουμε σε ένα DataFrame καλώντας τη read_csv() της Pandas:

```
df = pd.read_csv("winequality-white.csv", sep=";")
```

Όπως συμβαίνει με κάθε έργο, το πρώτο πράγμα που πρέπει να κάνουμε είναι να κατανοήσουμε τα δεδομένα μας.

28.1.3 Χαρακτηριστικά των δεδομένων και στόχος του project

Τα δεδομένα μας είναι σε μορφή CSV.

Αρχικά, ας εμφανίσουμε τον τύπο των δεδομένων για κάθε στήλη:

```
print(df.dtypes)
```

κι έτσι παίρνουμε:

```
= RESTART: C:/Users/NK/AppData/Lc
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

Δηλαδή:

- σταθερή οξύτητα – float64
- πτητική οξύτητα – float64
- κιτρικό οξύ – float64
- υπολειμματικά σάκχαρα– float64
- χλωρίδια – float64
- ελεύθερο διοξείδιο του θείου – float64
- ολικό διοξείδιο του θείου – float64
- πυκνότητα – float64
- pH – float64
- θειικά – float64
- alcohol – float64
- ποιότητα (προβλεπόμενη) – int64

Το σύνολο των δεδομένων μας αποτελείται από πολλά χαρακτηριστικά κρασιού τα οποία είναι όλα αριθμητικά και δεν υπάρχει πουθενά κάποια στήλη με κείμενο ή άλλο τύπο δεδομένων. Αυτό απλοποιεί το project μας και το κάνει κατάλληλο για ένα εισαγωγικό επίπεδο.

Συνοψίζοντας:

κάθε σειρά του συνόλου δεδομένων μας αντιπροσωπεύει ένα κρασί και κάθε στήλη περιέχει χαρακτηριστικά αυτού του κρασιού. Από αυτά τα χαρακτηριστικά, στόχος μας είναι να προβλέψουμε τη στήλη ποιότητας (quality) του κρασιού.

Ας εμφανίσουμε τώρα μια μικρή προεπισκόπηση των δεδομένων μας.

Για αυτό, χρησιμοποιούμε τη συνάρτηση head() που εμφανίζει την κεφαλή του συνόλου δεδομένων, τις πρώτες σειρές.

```
df.head(5)
```

Έξοδος:

```
= RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\wine-quality.py =
   fixed acidity  volatile acidity  citric acid  ...  sulphates  alcohol  qualit
y
0            7.0              0.27         0.36  ...       0.45      8.8
6
1            6.3              0.30         0.34  ...       0.49      9.5
6
2            8.1              0.28         0.40  ...       0.44     10.1
6
3            7.2              0.23         0.32  ...       0.40      9.9
6
4            7.2              0.23         0.32  ...       0.40      9.9
6

[5 rows x 12 columns]
```

Σ΄ αυτή την εκτύπωση τώρα, παρατηρούμε πως ενώ οι στήλες είναι 12, έχουμε πρόσβαση μόνο στις 5, οπότε χρειάζεται να δώσουμε μια επιπλέον εντολή στο pandas, για να μπορέσουμε να δούμε όλες τις στήλες (και πάλι μόνο τις 5 πρώτες σειρές. Η εντολή είναι:

```
pd.set_option('display.max_columns', None)
```

Η δεύτερη παράμετρος (None) μπορεί να πάρει τις τιμές int, ή None, για παρουσίαση συγκεκριμένου αριθμού στηλών, ή όλων (None). Έτσι, μια πλήρης απεικόνιση όλων των στηλών, μας δίνει την εκτύπωση:

```
= RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\wine-quality.py =
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol  quality
0      8.8        6
1      9.5        6
2     10.1        6
3      9.9        6
4      9.9        6
```

…όπου βλέπουμε και τις 12 στήλες, σε 3 ομάδες σειρών, με ξεχωριστά index numbers.

Εδώ μπορούμε πλέον να δούμε κάθε στήλη με το χαρακτηριστικό και κάποιες ενδεικτικές τιμές.

28.1.4 Διερευνητική Ανάλυση των Δεδομένων

Η διερευνητική ανάλυση των δεδομένων είναι μια φάση τεχνικής ανάλυσης ενός συνόλου δεδομένων. Επιτρέπει στους Επιστήμονες Δεδομένων και Μηχανικούς Μηχανικής Μάθησης να κατανοήσουν καλύτερα τα δεδομένα τους. Αλλά πάνω από όλα θα επιτρέψει και σε μας να προσδιορίσουμε το μοντέλο Μηχανικής Μάθησης που θα μας χρειαστεί γι’ αυτό το project.

Ας ξεκινήσουμε την ανάλυσή μας παρακάτω, προσπαθώντας να απλοποιήσουμε τους τρόπους ανάλυσης, εξάγοντας μια ερώτηση από κάθε τρόπο.

Μονομεταβλητή Ανάλυση (Univariate Analysis)

Η Μονομεταβλητή Ανάλυση είναι η διαδικασία επιθεώρησης κάθε χαρακτηριστικού ξεχωριστά.

Αυτό θα μας επιτρέψει να εμβαθύνουμε τις γνώσεις μας σχετικά με το σύνολο δεδομένων.

Εδώ είμαστε στη φάση της κατανόησης.

Το ερώτημα που σχετίζεται με την Μονομεταβλητή Ανάλυση είναι:

Ποια είναι τα δεδομένα που συνθέτουν το dataset μας;

Στόχος

Αριθμητικά ή Κατηγορικά Δεδομένα;

Πρώτα απ 'όλα, ας αναλύσουμε τα πιο σημαντικά δεδομένα για εμάς, τα δεδομένα της στήλης: quality.

Όταν νωρίτερα εμφανίσαμε τον τύπο της στήλης στόχου, παρατηρήσαμε ότι αποτελείται από ακέραιους αριθμούς.

Αυτό μπορεί να σημαίνει δύο πράγματα:

- είτε έχουμε άπειρο δυναμικό ακεραίων, από το 0 έως το άπειρο
- είτε ο χώρος δυνατοτήτων μας είναι περιορισμένος, για παράδειγμα από 0 έως 5

Για να βγάλουμε άκρη, θα πάρουμε τα δεδομένα ποιότητας και θα εμφανίσουμε τις μοναδικές τιμές (π.χ. αν έχουμε 50 σειρές με ποιότητα 8, θα εμφανίσουμε μία μόνο 8).

Εάν έχουμε μια πολύ μεγάλη λίστα, τότε μπορούμε να θεωρήσουμε ότι οι δυνατότητες είναι άπειρες, αλλά αν έχουμε μια μικρή λίστα, σημαίνει απαραίτητα ότι ο χώρος είναι περιορισμένος.

Οπότε, δίνοντας την εντολή:

```
df['quality'].unique()
```

παίρνουμε:

```
Έξοδος: πίνακας ([6, 5, 7, 8, 4, 3, 9])
```

Έτσι βλέπουμε πως έχουμε μια λίστα με επτά αριθμούς μεταξύ του 3 και του 9. Ως εκ τούτου, ο χώρος μας είναι περιορισμένος.

Αυτή είναι μια σημαντική πληροφορία για τη Μηχανική Μάθηση. Δεδομένου ότι έχουμε περιορισμένες δυνατότητες, θα χρησιμοποιήσουμε ένα μοντέλο classification. Διαφορετικά θα είχαμε χρησιμοποιήσει ένα μοντέλο regression.

Τα δεδομένα μας τώρα μας δείχνουν ότι μπορούμε να τα αναλύσουμε σαν κατηγορικές μεταβλητές

Τα Κατηγορικά Δεδομένα είναι δεδομένα που μπορούν να ομαδοποιηθούν σε συγκεκριμένες κατηγορίες.

Εδώ, μπορούμε, για παράδειγμα, να μιλήσουμε για κρασιά με ποιότητα 3. Αυτή είναι μια κατηγορία στο σύνολο δεδομένων μας.

Δεν θα μπορούσαμε να κάνουμε το ίδιο με τη στήλη χλωρίδια (chlorides) που αποτελείται από float.

28.1.5 Διανομή δεδομένων

Ας περάσουμε στην οπτική ανάλυση.

Όμως πριν να ξεκινήσουμε, θα χρειαστεί να εγκαταστήσουμε και να εισάγουμε ακόμα μια βιβλιοθήκη, τη seaborn. Έτσι λοιπόν, πρώτα έχουμε στο command line:

```
pip install seaborn
```

Αρχικά, εισάγουμε τις δύο κύριες βιβλιοθήκες για τη σχεδίαση γραφημάτων:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

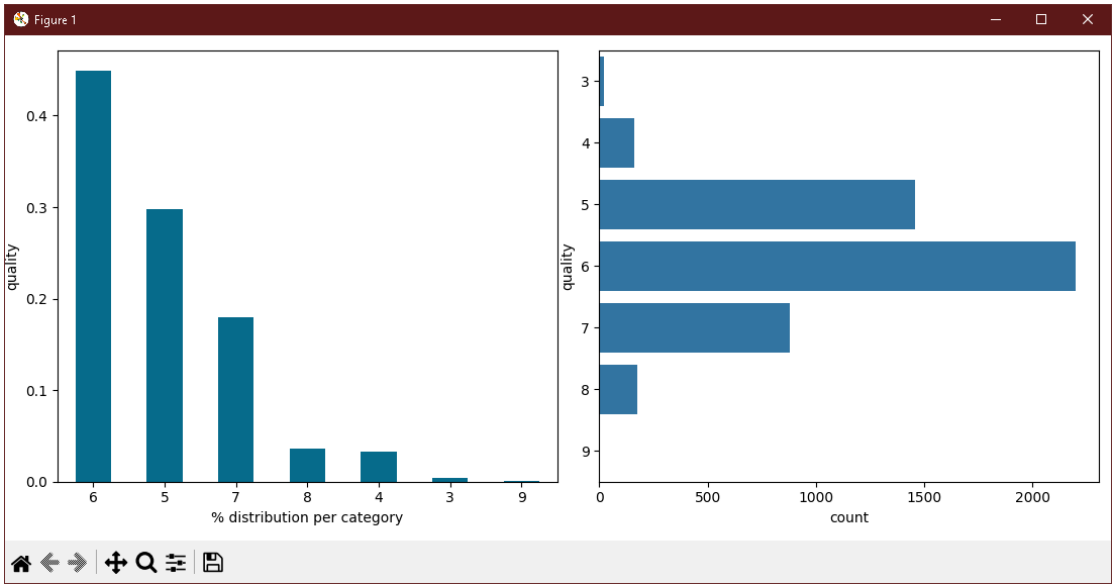
Στη συνέχεια, μπορούμε να εμφανίσουμε κάθε κατηγορίας ποιότητας κρασιού. Πόσες φορές παρουσιάζεται η κατηγορία 3, 4, 5, κ.λπ. στο σύνολο δεδομένων μας.

Θα δημιουργήσουμε δύο γραφήματα:

- Ένα κανονικοποιημένο γράφημα , το οποίο θα μας επιτρέψει να γνωρίζουμε την κατανομή ως ποσοστό
- Κι ένα κλασικό γράφημα , το οποίο θα μας επιτρέψει να γνωρίζουμε την ακατέργαστη κατανομή

```
plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
df['quality'].value_counts(normalize=True).plot.bar(rot
=0, color='#066b8b')
plt.ylabel('quality')
plt.xlabel('% distribution per category')
plt.subplot(1,2,2)
sns.countplot(data=df,y='quality')
plt.tight_layout()
plt.show()
```

κι έτσι έχουμε τα παρακάτω δύο γραφήματα



ΣΗΜΕΙΩΣΗ:

Τι κάνει το normalize;

Για παράδειγμα, αν στη στήλη ‘quality’ είχαμε τις τιμές [3, 4, 4, 5, 5, 5], τότε, καλώντας την df['quality'].value_counts(normalize=True) θα είχαμε μια εκτύπωση όπως η παρακάτω:

5 0.500000

4 0.333333

3 0.166667

Name: quality, dtype: float64

Σ’ αυτή την περίπτωση, βλέπουμε ότι το 50% των τιμών είναι ίσες με 5, 33.33% ίσες με 4, και 16.67% ίσες με 3. Αυτή η εικόνα μας δίνει μια καλύτερη αίσθηση της κατανομής των τιμών της ποιότητας με μια κανονικοποιημένη φόρμα. Αυτό μας είναι χρήσιμο για να δημιουργούμε οπτικοποιήσεις, ή να πραγματοποιούμε αναλύσεις όπου οι σχετικές αναλογίες μας δίνουν περισσότερες πληροφορίες από τις τιμές.

Και στα δύο γραφήματα βλέπουμε την ίδια κατανομή που είναι κανονική.

Πρώτα απ' όλα, παρατηρούμε ότι τα κρασιά ποιότητας 6 αντιπροσωπεύονται περισσότερο στο σύνολο δεδομένων μας. Στα αριστερά, μπορούμε να δούμε ότι αντιπροσωπεύουν περισσότερο από το 40% του συνόλου των δεδομένων μας (πλησιάζουμε το 50%) και στα δεξιά ότι αντιπροσωπεύουν περίπου 2500 γραμμές (άρα 2500 κρασιά).

Γενικότερα, μπορούμε να παρατηρήσουμε ένα πράγμα που ίσως θα χαρακτηρίζαμε αρνητικό: ότι το σύνολο των δεδομένων μας δεν κατανέμεται ισόποσα.

Αυτή η ανισότητα μπορεί να επηρεάσει την απόδοση του μοντέλου που θα καθορίσουμε. Πράγματι, υπάρχουν τόσο λίγα κρασιά ποιότητας 9 που το μοντέλο δεν θα είναι σε θέση να αναλύσει τα χαρακτηριστικά που τα διαφοροποιούν από τα άλλα.

Αλλά από την άλλη πλευρά, η διαφορά στην κατανομή ενός συνόλου δεδομένων είναι κάτι φυσιολογικό που συμβαίνει συχνά σε έργα του πραγματικού κόσμου. Είναι ένα μειονέκτημα που πρέπει να αντιμετωπίσουν οι Data Scientists / Machine Learning Engineers.

28.1.6 Χαρακτηριστικά – Αριθμητικά δεδομένα

Διανομή και Box plot

Αναλύουμε τώρα τα χαρακτηριστικά των κρασιών. Αυτά είναι τα δεδομένα που θα επιτρέψουν στο μοντέλο Machine Learning να ανιχνεύσει την ποιότητα των κρασιών.

Αρχικά, μπορούμε να δημιουργήσουμε ένα δευτερεύον DataFrame που περιέχει μόνο τα δεδομένα μας, δηλαδή, να αφαιρέσουμε τη στήλη της ποιότητας:

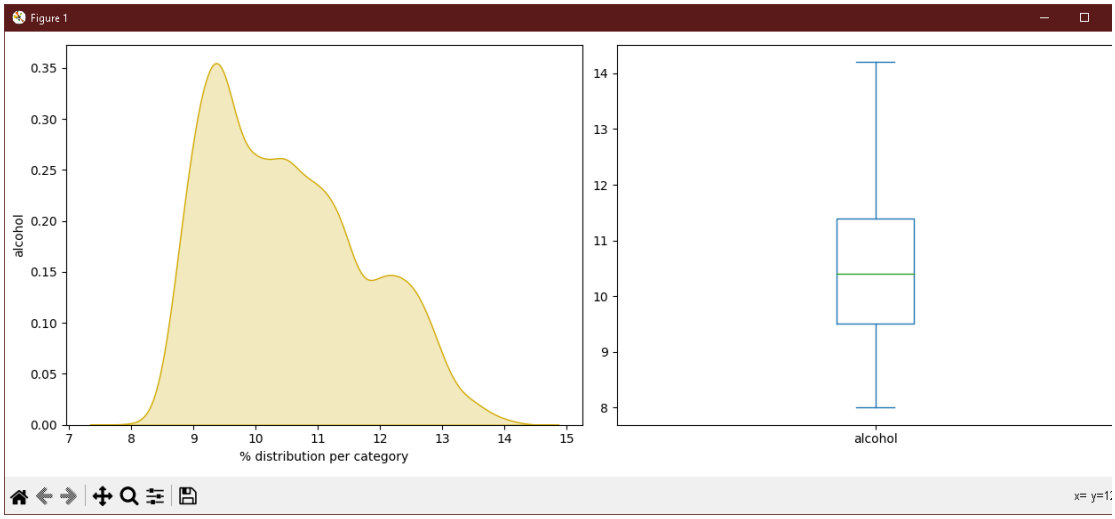
```
df_features = df.drop(columns='quality')
```

Τώρα ας αναλύσουμε τη στήλη ‘alcohol’ που αντιπροσωπεύει τον βαθμό αλκοόλης του κρασιού.

Εμφανίζουμε εδώ την κατανομή σε ένα κλασικό γράφημα και σε ένα διάγραμμα πλαισίου:

```
plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
ax =
sns.kdeplot(df_features['alcohol'],shade=True,color='#d
1aa00')
plt.ylabel('alcohol')
plt.xlabel('% distribution per category')
plt.subplot(1,2,2)
df_features['alcohol'].plot.box()
plt.tight_layout()
plt.show()
```

και παρατηρούμε τα γραφήματά μας:



Αριθμητικά δεδομένα

Θέλουμε να μάθουμε αν υπάρχει κάποια ανωμαλία στο γράφημά μας:

- ασυνήθιστη κατανομή
- δεδομένα που λείπουν
- μοναδική τιμή πολύ μακριά από τον μέσο όρο

Το γράφημα φαίνεται αρκετά φυσιολογικό, σε αντίθεση με τα δεδομένα που αναλύθηκαν σε προηγούμενο project μας για τις δασικές πυρκαγιές.

28.1.7 Ομαδοποιημένη ανάλυση

Εδώ, θα εμφανίσουμε τα ίδια γραφήματα όπως πριν, αλλά για όλες τις στήλες μας.

Για να απλοποιήσουμε τα πράγματα, θα δημιουργήσουμε έναν βρόχο “for”.

Για να το κάνουμε αυτό παίρνουμε τα ονόματα των στηλών μας:

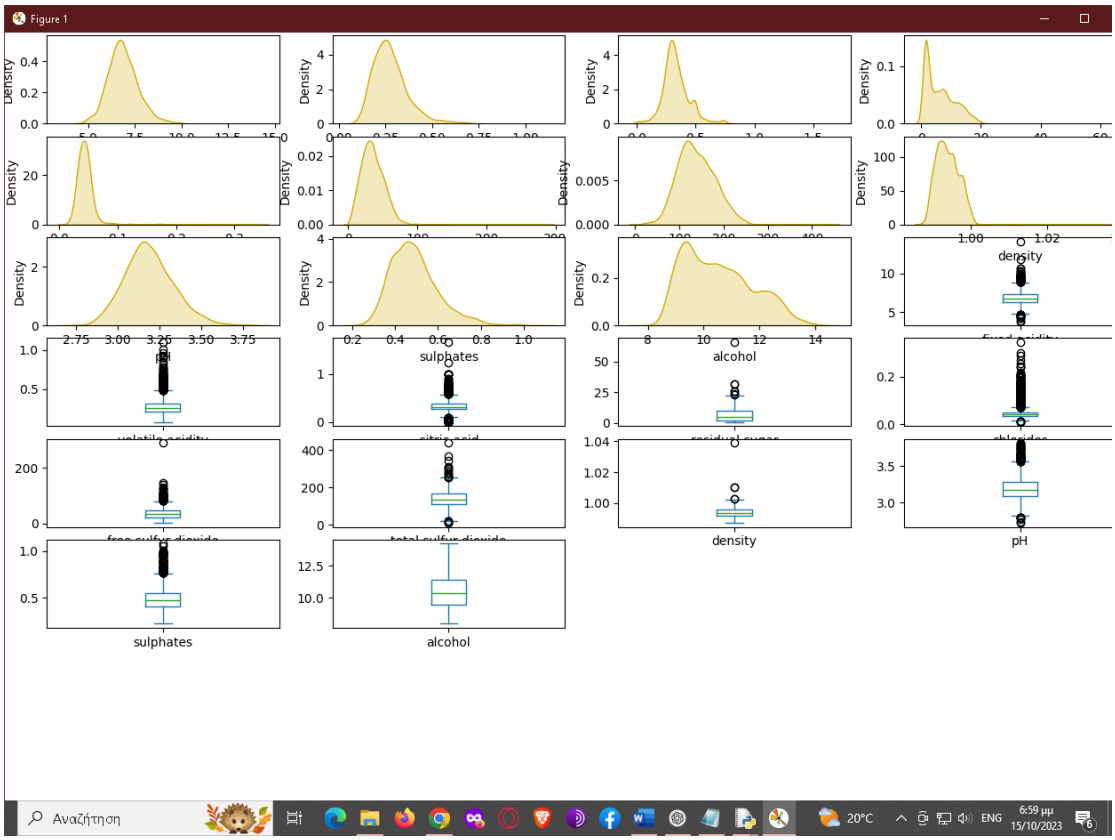
```
num_columns = df_features.columns.tolist()
num_columns
```

Έξοδος:

```
= RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\wine-quality.py =
['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chloride
s', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates',
 'alcohol']
```

Στη συνέχεια περνάμε από κάθε μία από αυτές τις στήλες για να εμφανίσουμε τα γραφήματα μας (κλασική κατανομή και διάγραμμα πλαισίου):

```
plt.figure(figsize=(18,40))
for i,col in enumerate(num_columns,1): # 1 = optional
starting number
     plt.subplot(8,4,i)
     sns.kdeplot(df[col],color='#d1aa00',shade=True)
     plt.subplot(8,4,i+11)
     df[col].plot.box()
plt.tight_layout()
plt.show()
```



Αν και το γράφημα είναι αρκετά μεγάλο και περιέχει πολλά subplots, δεν φαίνεται να περιέχεται κάποια «άκυρη» τιμή.

Μπορούμε βέβαια πάντα να χρησιμοποιούμε πιο απλοποιημένα γραφήματα με λιγότερα ζητούμενα, όμως, θα χρειαστεί εδώ να δούμε δύο μετρήσεις, τις “skewness” και “kurtosis” οι οποίες μας επιτρέπουν να καταλαβαίνουμε την κατανομή των δεδομένων με μεγαλύτερη

σαφήνεια (όποιος ενδιαφέρεται, μπορεί εδώ να διαβάσει ολόκληρο άρθρο για τα δύο αυτά είδη μετρήσεων).

Εμφανίζουμε λοιπόν αυτές τις δύο μετρήσεις για καθεμία από τις στήλες των χαρακτηριστικών μας (ορίζουμε και πάλι την εκτύπωση των μέγιστων στηλών). Ας ζητήσουμε μια εκτύπωση:

```
result_df =
pd.DataFrame(data=[df[num_columns].skew(),df[num_column
s].kurtosis()],index=['skewness','kurtosis'])
pd.set_option('display.max_columns', None)

print(result_df)
```

Στρεπτότητα & Κύρτωση

Κι έχουμε έξοδο:

```
= RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\wine-quality.py =
          fixed acidity  volatile acidity  citric acid  residual sugar  \
skewness       0.647751          1.576980     1.281920        1.077094
kurtosis       2.172178          5.091626     6.174901        3.469820

          chlorides  free sulfur dioxide  total sulfur dioxide   density  \
skewness   5.023331             1.406745              0.390710  0.977773
kurtosis  37.564600            11.466342              0.571853  9.793807

                pH  sulphates   alcohol
skewness  0.457783   0.977194  0.487342
kurtosis  0.530775   1.590930 -0.698425
```

Παρατηρούμε μια ασυνήθιστη διαφορά στα χλωρίδια με κύρτωση 37 και στρεπτότητα 5, στο ελεύθερο διοξείδιο του θείου με κύρτωση 11. Τα πιο ισορροπημένα δεδομένα είναι το pH και το αλκοόλ.

Διαπιστώνεται ότι υπάρχουν ακραίες τιμές, δεδομένα με τιμές πολύ μακριά από τον μέσο όρο, για τις στήλες χλωρίδια, ελεύθερο διοξείδιο του θείου, πυκνότητα, κιτρικό οξύ και πτητική οξύτητα.

Στις περισσότερες κατανομές, είναι φυσιολογικό να υπάρχουν ακραίες τιμές.

Αλλά οι ακραίες τιμές δεν είναι συχνές, ακόμη και μη φυσιολογικές. Κάποιες φορές είναι σφάλμα στο σύνολο δεδομένων.

28.1.8 Ανάλυση Διμεταβλητών - Bivariate Analysis

Έχουμε πλέον κατανοήσει την κατανομή των δεδομένων μας χάρη στη Μονομεταβλητή Ανάλυση. Η ιδέα τώρα είναι να συνεχίσουμε αυτή την ανάλυση εντοπίζοντας τους πιθανούς δεσμούς μεταξύ των χαρακτηριστικών μας και του στόχου μας, της ποιότητας των κρασιών.

Η Διμεταβλητή Ανάλυση είναι η ανάλυση του καθενός από τα χαρακτηριστικά, τοποθετώντας το σε σχέση με τον στόχο μας.

Αυτό θα μας επιτρέψει να κάνουμε υποθέσεις σχετικά με το σύνολο των δεδομένων.

Εδώ βρισκόμαστε στη φάση της θεωρίας.

Το ερώτημα που σχετίζεται με την Ανάλυση Διμεταβλητών είναι:
Υπάρχει σύνδεση μεταξύ των χαρακτηριστικών μας και του στόχου;

Για παράδειγμα, θα μπορούσε να υπάρχει ένας σύνδεσμος όπως: όσο μεγαλύτερη είναι η πυκνότητα του κρασιού, τόσο χαμηλότερη είναι η ποιότητα.

Αν η σχέση είναι τόσο προφανής, μπορούμε να προβλέψουμε άμεσα την ποιότητα του κρασιού κοιτάζοντας μόνο την πυκνότητά του.

Αλλά εάν οι σύνδεσμοι είναι πιο περίπλοκοι και λιγότερο σίγουροι, θα χρειαστεί ένα πολύπλοκο μοντέλο Μηχανικής Μάθησης για να πετύχουμε τον στόχο μας.

28.1.9 Αριθμητικά δεδομένα

Γράφημα βιολιού

Είδαμε ότι τα δεδομένα χαρακτηριστικών μας είναι αριθμητικά δεδομένα και τα δεδομένα στόχου μας είναι κατηγορικά δεδομένα.

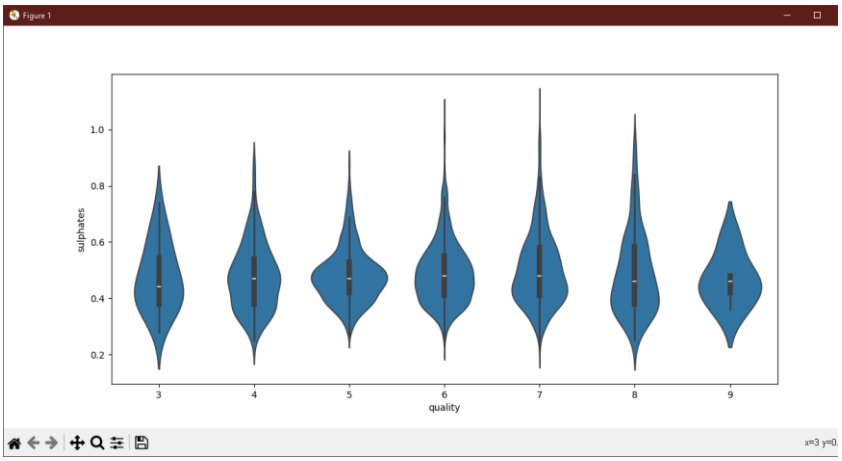
Αυτό μας διευκολύνει να κάνουμε την ανάλυσή μας.

Ξεκινάμε με την ανάλυση του χαρακτηριστικού sulphate. Μελετώντας τα δεδομένα θέλουμε να απαντήσουμε στο παρακάτω ερώτημα. Υπάρχει σχέση μεταξύ της ποσότητας των θειικών αλάτων και της ποιότητας του κρασιού;

```
plt.figure(figsize=(16,6))
sns.violinplot(data=df, x='quality', y='sulphates')
```

Γράφημα βιολιού

Έξοδος:



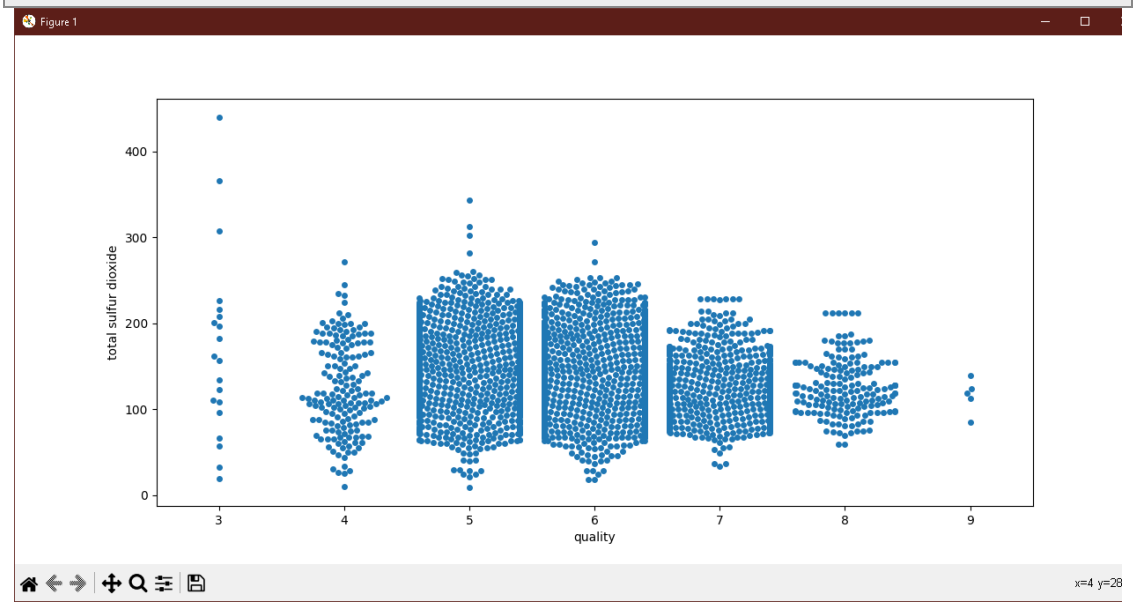
Η διανομή φαίνεται να είναι ίδια για κάθε ποιότητα κρασιού. Εδώ, θέλουμε να βρούμε ένα μοτίβο που θα μας βοηθούσε να προβλέψουμε την ποιότητα του κρασιού.

Για παράδειγμα, θα ήταν ενδιαφέρον αν μπορούσαμε να δούμε ότι το μέσο επίπεδο των sulfate αυξάνεται με την ποιότητα του κρασιού. Εδώ, αντίθετα, βλέπουμε ότι ο μέσος όρος παραμένει ίδιος για όλες τις κατηγορίες. Υπάρχουν κάποια ακραία δεδομένα για τις ποιότητες 6, 7 και 8. Αλλά τίποτα αξιοσημείωτο ή πραγματικά χρήσιμο για να δείξει μια τάση. Ας περάσουμε σε άλλο plot.

28.1.10 Γράφημα Σμήνους

Ας εμφανίσουμε τώρα το total sulfur dioxide σε ένα γράφημα σμήνους και σε συνάρτηση με την ποιότητα του κρασιού:

```
plt.figure(figsize=(16,6))
sns.swarmplot(x="quality", y="total sulfur dioxide",
data=df)
```



Γράφημα Σμήνους

Εδώ πρέπει να προσέξουμε δύο πράγματα.

Πρώτα βλέπουμε ότι υπάρχουν πολύ περισσότερα σημεία στα δεδομένα 5, 6 και 7.

Αυτό δεν είναι τάση. Αντίθετα, αυτό είναι μια απόκλιση στο σύνολο των δεδομένων μας.

Ας θυμηθούμε ότι, παραπάνω, είδαμε ότι υπήρχαν περισσότερα δεδομένα για τις ποιότητες κρασιού 5, 6 και 7 και πολύ λιγότερα για τα δεδομένα 3 και 9.

Αυτό βλέπουμε κι εδώ και τώρα καταλαβαίνουμε πως η Μονομεταβλητή Ανάλυση δείχνει τη σημασία της γιατί μας επιτρέπει να εντοπίσουμε τις αποκλίσεις στην αρχή της ανάλυσής μας.

Πράγματι, θα μπορούσε κανείς να σκεφτεί ότι αυτή η ανώμαλη κατανομή είναι συγκεκριμένη για το χαρακτηριστικό total sulfur dioxide. Στην πραγματικότητα, αυτή η κατανομή είναι αναπόσπαστο μέρος του συνόλου δεδομένων μας και η απόκλιση που είδαμε στη συνολική ανάλυση αντικατοπτρίζεται εδώ στην πιο λεπτομερή ανάλυση.

Δεύτερον, μπορούμε να παρατηρήσουμε μια τάση: όσο αυξάνεται η ποιότητα του κρασιού, τόσο λιγότερο είναι το χαρακτηριστικό total sulfur dioxide.

Η τάση είναι αδύναμη αλλά υπάρχει.

Ωστόσο, δύο πράγματα πρέπει να σημειωθούν:

- Τα δεδομένα φαίνεται να αλλάζουν σημαντικά μετά την ποιότητα του κρασιού 4

- τα δεδομένα για το 3 και το 9 είναι πολύ αραιά. Τι θα γινόταν αν το σύνολο δεδομένων είχε συμπληρωθεί με περισσότερες 3 και 9 ποικιλίες κρασιού; Θα βλέπαμε την τάση να αντιστρέφεται;

Αυτά τα ερωτήματα θα παραμείνουν αναπάντητα επειδή το σύνολο δεδομένων είναι ως έχει. Και πάλι, αυτό είναι ένα επαναλαμβανόμενο φαινόμενο σε σύνολα δεδομένων του πραγματικού κόσμου και είναι σημαντικό να γνωρίζουμε πώς να το αντιμετωπίσουμε (αλλά και να ξέρουμε πότε να ζητήσετε από τον πελάτη περισσότερα δεδομένα).

28.1.11 Δουλεύοντας με τις αποκλίσεις

Ας δούμε τώρα μια τεχνική για την αξιολόγηση αυτής της δυνατότητας παρά την έλλειψη δεδομένων.

Πρώτα, υποθέτουμε ότι το σύνολο δεδομένων μας έχει μια κατανομή αντιπροσωπευτική του πραγματικού κόσμου.

Αυτό σημαίνει ότι υποθέτουμε ότι αν προσθέταμε χιλιάδες κρασιά στο σύνολο δεδομένων μας, ο μέσος όρος αυτών του total sulfur dioxide που θα προέκυπτε θα ήταν ο ίδιος με αυτόν που έχουμε τώρα.

Ως εκ τούτου, τώρα μπορούμε να αναλύσουμε τον τρέχοντα μέσο όρο μας και να θεωρήσουμε ότι το αποτέλεσμα που προκύπτει θα είναι αντιπροσωπευτικό του πραγματικού κόσμου.

Υπολογίζουμε τον μέσο όρο total sulfur dioxide για κάθε ποιότητα κρασιού:

```
quality_cat = df.quality.unique()
quality_cat.sort()
qual_TSD = []
for i,quality in enumerate(quality_cat):
```

```
  qual_TSD.append([quality, df['total sulfur
dioxide'].loc[df['quality'] == quality].mean()])
```

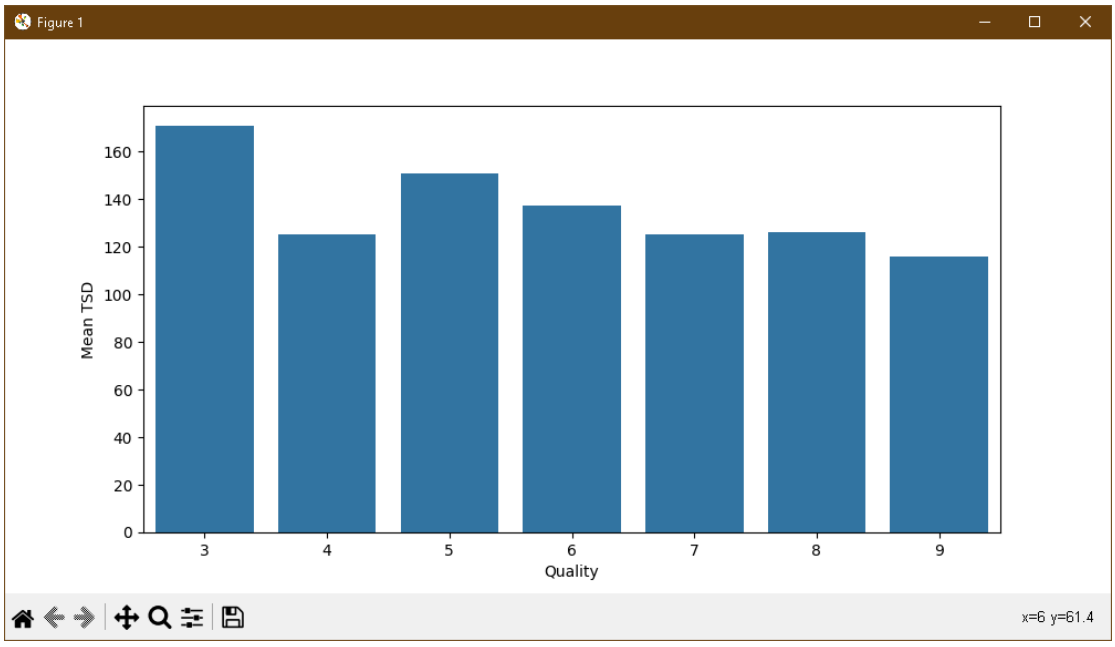
Λαμβάνουμε 7 γραμμές, και καθεμία από αυτές έχει το μέσο όρο που αντιστοιχεί στο total sulfur dioxide σε κάθε επίπεδο ποιότητας.

Ας βάλουμε αυτά τα δεδομένα σε ένα νέο DataFrame:

```
df_qual_TSD = pd.DataFrame(qual_TSD, columns
=['Quality', 'Mean TSD'])
```

Και τώρα ας αναλύσουμε τον μέσο όρο total sulfur dioxideγια κάθε επίπεδο ποιότητας:

```
plt.figure(figsize=(10,5))
sns.barplot(x="Quality", y="Mean TSD",
data=df_qual_TSD)
plt.show()
```



Οριζόντιο ραβδόγραμμα

Η τάση που παρατηρήσαμε πριν υπάρχει στο σύνολο δεδομένων μας.

Όσο ο μέσος όρος του total sulfur dioxide μειώνεται, τόσο αυξάνεται η ποιότητα του κρασιού, με εξαίρεση την ποιότητα 4.

Ας συνεχίσουμε την Ανάλυση Διμεταβλητών!

28.1.12 Κατηγορική μέθοδος σε αριθμητικά δεδομένα

Μεταμορφώνοντας τα δεδομένα μας

Ας δούμε τώρα μια άλλη προσέγγιση για την ανάλυση των δεδομένων μας.

Τα τρέχοντα δεδομένα μας είναι αριθμητικά. Δηλαδή δεν μπορούμε να τα κατατάξουμε σε κατηγορίες. Τουλάχιστον όχι άμεσα.

Αυτό που προτείνουμε εδώ είναι να μετατρέψουμε τα αριθμητικά μας δεδομένα σε κατηγορίες.

Ας πάρουμε τη στήλη ‘alcohol’ που αντιπροσωπεύει τον βαθμό αλκοόλης κάθε κρασιού.

Για να μετατρέψουμε αυτές τις πληροφορίες σε κατηγορία, θα επιλέξουμε τιμές για να τις ταξινομήσουμε.

Θα πάρουμε όλα τα κρασιά με περιεκτικότητα σε αλκοόλ μικρότερη από 9,5° και θα τα βάλουμε στην κατηγορία Χαμηλή. Το ίδιο θα κάνουμε έτσι και για τις 4 κλάσεις παρακάτω:

- Χαμηλή όταν είναι μικρότερη από 9,5°.
- Μέτρια μεταξύ 9,5° και 11°.
- Υψηλή μεταξύ 11° και 12,5°.
- Πολύ υψηλή όταν είναι υψηλότερη από 12,5°.

Φροντίζουμε κάθε κατηγορία, να έχει διαφορά 1,5° βαθμού. Αυτός ο διαχωρισμός είναι αυθαίρετος. Θα μπορούσαμε να έχουμε επιλέξει τα τεταρτημόρια που υποδεικνύονται από το διάγραμμα πλαισίου στη μονομεταβλητή ανάλυση.

```
def alcohol_cat(alcohol):
    if alcohol <= 9.5:
        return "Low"
    elif alcohol <= 11:
        return "Moderate"
    elif alcohol <= 12.5:
        return "High"
    else:
        return "Very High"

df['alcohol_category'] =
df['alcohol'].apply(alcohol_cat)
df.sample(frac=1).head()
```

Τώρα, έχουμε τη νέα στήλη / κατηγορία alcohol_category που αντιπροσωπεύει τον βαθμό αλκοόλ ως κατηγορικό δεδομένο.

28.1.13 Εμφάνιση των δεδομένων μας

Τώρα θα θέλαμε να συσχετίσουμε αυτά τα δεδομένα με την ποιότητα του κρασιού.

Για αυτό χρησιμοποιήσουμε ένα κανονικοποιημένο Crosstab (αντικείμενο Pandas) που εμφανίζουμε ως γραφική παράσταση ράβδων.

```
import numpy as np

plt.figure(figsize=(15,30))
```

```
cross =
pd.crosstab(index=df['quality'],columns=df['alcohol_cat
egory'],normalize='index')
cross.plot.barh(stacked=True,rot=40,cmap='crest_r').leg
end(bbox_to_anchor=(1.0, 1.0))
plt.xlabel('% distribution per category')
plt.xticks(np.arange(0,1.1,0.1))
plt.title("Wine Quality each
{}".format('alcohol_category'))
plt.show()
```

Κατηγορικά x Κατηγορικά Δεδομένα

Σε αυτό το γράφημα, μπορούμε να δούμε την κατηγορία αλκοολούχων ποτών που συνθέτουν κάθε κατηγορία ποιότητας κρασιού μας.

Παρατηρούμε ότι δεν υπάρχουν κρασιά χαμηλού αλκοόλ στα κρασιά ποιότητας 9.

Γενικά, μπορούμε να δούμε μια προφανή τάση: όσο αυξάνεται η περιεκτικότητα σε αλκοόλ, αυξάνεται και η ποιότητα του κρασιού.

Αυτό δεν σημαίνει ότι ένα κρασί 14° είναι απαραίτητα καλό. Αλλά μάλλον ότι αν πάρετε ένα τυχαίο κρασί 14°, υπάρχει μεγάλη πιθανότητα να είναι ένα κρασί εξαιρετικής ποιότητας.

28.1.14 Βήμα δεύτερο: Μηχανική μάθηση

Έχουμε πλέον ενημερωθεί αρκετά για τα δεδομένα μας και προχωράμε σε περαιτέρω προετοιμασία τους.

Έχουμε ήδη ένα DataFrame για τις δυνατότητές μας, το df_features, ας πάρουμε ακόμα ένα για τον στόχο μας το οποίο θα ονομάσουμε df_label:

```
df_label = df['quality']
```

Τώρα χωρίζουμε τα δεδομένα μας σε διαφορετικά σύνολα. Το ένα για εκπαίδευση, και το άλλο για δοκιμή της απόδοσής του.

Ονομάζουμε τα χαρακτηριστικά X και την ετικέτα y(στόχος).

Αυτό μας δίνει τέσσερα σύνολα δεδομένων:

X_train

X_test

y_train

y_test

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(df_features, df_label, test_size=0.20)
```

Τα διαχωρίσαμε έτσι ώστε να έχουμε το 80% των συνολικών δεδομένων μας στο X και το 20% στο y. Ελέγχουμε ότι τα σετ έχουν το σωστό μέγεθος:

```
print((len(X_train), len(y_train)))
print((len(X_test), len(y_test)))
```

Έξοδος:

(3918, 3918)

(980, 980)

Με αυτά τα δεδομένα μπορούμε να εκπαιδεύσουμε μοντέλα μηχανικής μάθησης.

28.1.15 Τι είναι η εκπαίδευση στη Μηχανική Μάθηση;

Ας δούμε πώς λειτουργεί η εκπαίδευση ενός μοντέλου:

- Το μοντέλο μηχανικής μάθησης εξετάζει κάθε τιμή των X_train και y_train.
- Καθιερώνει μια συνάρτηση που επιτρέπει, με τις τιμές X_train, να βρούμε το y_train
- Δοκιμάζουμε την απόδοση του μοντέλου βλέποντας εάν, από το X_test, το μοντέλο καταφέρνει να συναγάγει το y_test
- Με άλλα λόγια, το μοντέλο είναι εκπαιδευμένο να συνάγει την ποιότητα του κρασιού από το 80% των φιαλών και τα χαρακτηριστικά τους.

Η εκπαίδευση ολοκληρώνεται όταν το μοντέλο έχει δημιουργήσει μια εξίσωση που συσχετίζει τα μπουκάλια κρασιού με την αντίστοιχη ποιότητά τους.

Στη συνέχεια, μπορεί να δοκιμάσει την απόδοσή του στο υπόλοιπο 20% των φιαλών.

Στόχος του είναι να βρει μια συνάρτηση που λειτουργεί τόσο για τα δεδομένα εκπαίδευσης όσο και για τα δεδομένα δοκιμής και εδώ είναι που βρίσκεται η δυσκολία.

Εάν τα δεδομένα εκπαίδευσης αντιπροσωπεύουν τον πραγματικό κόσμο, τότε το μοντέλο θα έχει καλή απόδοση.

Αντίθετα, εάν τα δεδομένα δεν αντιπροσωπεύουν την ποικιλομορφία των κρασιών, τότε θα έχει μεγαλύτερη δυσκολία στην πρόβλεψη των δεδομένων δοκιμής.

Πράγματι, εάν το μοντέλο δεν δει ποτέ ένα κρασί ποιότητας 9 κατά τη διάρκεια της προπόνησης, κατά τη διάρκεια του τεστ απόδοσης, εάν συναντήσει ένα μπουκάλι ποιότητας 9, δεν θα μπορεί να το ταξινομήσει με τον σωστό τρόπο.

Εδώ εμφανίζεται η σημασία της Μονομεταβλητής Ανάλυσης, καθώς μας επιτρέπει να κατανοήσουμε τις αποκλίσεις στο σύνολο δεδομένων μας και επομένως τα πιθανά εμπόδια που θα αντιμετωπίσουμε.

Ας προχωρήσουμε στα Μοντέλα Μηχανικής Μάθησης.

28.1.16 Logistic Regression

Η Logistic Regression είναι ένα μοντέλο που μας επιτρέπει να συμπεράνουμε μια κατηγορική μεταβλητή από δεδομένα των χαρακτηριστικών.

Αυτό το μοντέλο αναλύει τα δεδομένα μας ένα προς ένα για να καθορίσει έναν απλό κανόνα για τον προσδιορισμό της κατηγορίας στην οποία ανήκει κάθε κρασί.

Με μαθηματικούς όρους, η Logistic Regression δημιουργεί μια συνάρτηση που επιτρέπει, από δεδομένα χαρακτηριστικών, να υπολογίσει την πιθανότητα να ανήκει σε κάθε κλάση.

```
from sklearn.linear_model import LogisticRegression

logisticRegression = LogisticRegression()
logisticRegression.fit(X_train, y_train)
```

Και η βαθμολογία που λαμβάνουμε είναι:

```
logisticRegression.score(X_test, y_test)
```

Όμως, ένα πρακτικό θέμα που προκύπτει, είναι ότι, λόγω διαφοράς μηχανημάτων (υπολογιστών), αλλά και εκδόσεων του λογισμικού, στην προκειμένη περίπτωση, ο κώδικάς μας δεν τρέχει λόγω του παρακάτω warning καθώς και λαθών:

```
Warning (from warnings module):
  File "C:\Users\NK\AppData\Local\Programs\Python\Python311\Lib\site-packages\sk
learn\linear_model\_logistic.py", line 460
    n_iter_i = _check_optimize_result(
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n
```

Αν δούμε τα σφάλματα, διαπιστώνουμε ότι χρειάζεται να αυξήσουμε τον αριθμό των επαναλήψεων περασμάτων, ώστε να οδηγηθούμε σε ένα αποτέλεσμα.

Έτσι, τελικά τρέχουμε τον παρακάτω κώδικα:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.preprocessing import StandardScaler  #
Import the StandardScaler

df = pd.read_csv("winequality-white.csv", sep=";")

df_features = df.drop(columns='quality')

df_label = df['quality']

X_train, X_test, y_train, y_test =
train_test_split(df_features, df_label, test_size=0.20)

# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Increase the number of iterations
logisticRegression = LogisticRegression(max_iter=1000)
# Increase the max_iter
logisticRegression.fit(X_train, y_train)

score = logisticRegression.score(X_test, y_test)
print("Model Score:", score)
```

κι έτσι, έχουμε την παρακάτω εκτύπωση:

```
= RESTART: C:/Users/NK/AppData/L
Model Score: 0.5673469387755102
```

Έχουμε δηλαδή ακρίβεια περίπου 57% που δεν είναι πολύ υψηλή.

Ας δούμε πώς γίνεται η πρόβλεψη της κατηγορίας του πρώτου κρασιού στο σύνολο δεδομένων μας.

Χρησιμοποιούμε τη συνάρτηση predict_proba() που μας επιτρέπει να εμφανίζουμε την πιθανότητα το εν λόγω κρασί να ανήκει σε κάποιο επίπεδο ποιότητας κρασιού:

Ας εμφανίσουμε τις πιθανότητες να ανήκει σε κάθε κλάση για την πρώτη σειρά του συνόλου δεδομένων μας:

```
predicted_prob =
logisticRegression.predict_proba(X_test.iloc[:1])
print(predicted_prob)
```

Έξοδος:

```
Predicted Probabilities for the First Sample:
[[4.63540890e-03 6.00108787e-03 2.01134898e-01 5.58557946e-01
  1.91042212e-01 3.83246603e-02 3.03787198e-04]]
```

Δηλαδή: Κλάση : Πιθανότητα

3 : 4,6354

4 : 6,0010

5 : 2,0113

6 : 5,5855

7 : 1,9104

8 : 3,8324

9 : 3,0378

Εδώ η υψηλότερη πιθανότητα είναι για την κλάση 4.

Το μοντέλο μας προβλέπει ότι το κρασί είναι ποιότητας 4.

Ποιο είναι όμως το σωστό αποτέλεσμα;

```
Print(y_test.iloc[:1])
```

```
4852    5
Name: quality, dtype: int64
```

Στην πραγματικότητα το κρασί είναι ποιότητας 5. Ας δούμε άλλα μοντέλα.

28.1.17 Support Vector Machines

Ο Support Vector Machines είναι ένας αλγόριθμος μηχανικής μάθησης υψηλής απόδοσης.

Ο SVM σχεδιάζει έναν κενό χώρο (ας φανταστούμε έναν 2D χώρο με τετμημένη και τεταγμένη). Στη συνέχεια εξετάζει κάθε σειρά του συνόλου δεδομένων μας και του δίνει μια θέση στο χώρο σύμφωνα με τα χαρακτηριστικά του (X) αλλά και την κατηγορία (y) στην οποία ανήκει.

Όταν μια σειρά ανήκει σε μια νέα κατηγορία (μια διαφορετική ποιότητα κρασιού), προβάλλει αυτό το σημείο για να μεγιστοποιήσει την απόσταση μεταξύ αυτής της κατηγορίας και των προηγούμενων.

Το μοντέλο επαναλαμβάνει αυτή τη λειτουργία μέχρι να αποκτήσει έναν πλήρη χώρο που ονομάζεται χάρτης SVM.

Για να προβλέψει, ο SVM απλώς παίρνει τα νέα δεδομένα και ανάλογα με τα χαρακτηριστικά, τοποθετεί το σχετικό σημείο του στο χώρο. Η θέση του θα δώσει στη συνέχεια την κατηγορία στην οποία ανήκει.

Πάμε να δούμε τον SVM στην πράξη:

```
from sklearn import svm

SVM = svm.SVC()
SVM.fit(X_train, y_train)
```

```
# Και η βαθμολογία που λαμβάνουμε:
svm_score = SVM.score(X_test, y_test)
print(svm_score)
```

```
= RESTART: C:/Users
0.576530612244898
```

Ίδια ακρίβεια με την Logistic Regression. Ας συνεχίσουμε με το επόμενο μοντέλο!

28.1.18 Stochastic Gradient Descent

Αν έχουμε κατανοήσει πώς λειτουργεί ο SVM, θα καταλάβουμε εύκολα και το Stochastic Gradient Descent.

Ο SVM αποτελούνταν από τη λήψη σημείων και την προβολή τους σε ένα χώρο, ένα επίπεδο, με τέτοιο τρόπο ώστε να μεγιστοποιείται η απόσταση μεταξύ των διαφορετικών κατηγοριών.

Με μαθηματικούς όρους, αυτό ονομάζεται εύρεση του βέλτιστου μιας συνάρτησης.

Και το Stochastic Gradient Descent (SGD) είναι στην πραγματικότητα ένας βελτιστοποιητής που επιτρέπει τη βελτίωση αλγορίθμων (ή συναρτήσεων) όπως ο SVM, ο Logistic Regression, κ.λπ.

Το ίδιο εργαλείο βελτιστοποίησης χρησιμοποιείται στα νευρωνικά δίκτυα. Αυτό καθιστά το SGD μία από τις θεμελιώδεις έννοιες της Μηχανικής Μάθησης.

Στη χρήση του τώρα, από προεπιλογή βελτιστοποιεί ένα SVM:

```
from sklearn.linear_model import SGDClassifier

SGD = SGDClassifier()
SGD.fit(X_train, y_train)
# Και η βαθμολογία που λαμβάνουμε είναι:
sgd_score = SGD.score(X_test, y_test)
```

```
print(sgd_score)
```

Η έξοδός μας είναι:

```
= RESTART: C:/Users/N
0.47959183673469385
|
```

47%, πιο χαμηλά ακόμα από τους προηγούμενους αλγορίθμους.

Ακόμα δεν φαίνεται να ταιριάζει με το σύνολο δεδομένων μας. Ας προχωρήσουμε σε ακόμα ένα μοντέλο!

28.1.19 Δέντρο απόφασης

Το Decision Tree είναι ένας από τους πιο ευρέως χρησιμοποιούμενους αλγόριθμους και ο πιο εύκολος στην κατανόηση!

Κάνει υποθέσεις με τη μορφή δέντρου. Κάθε κόμβος του δέντρου αντιπροσωπεύει μια υπόθεση για ένα χαρακτηριστικό.

Το δέντρο μπορεί να έχει τεράστιο μήκος και μπορεί να δοκιμάσει πολλά χαρακτηριστικά με διαφορετικούς τρόπους.

Ας δούμε πως μπορούμε να χρησιμοποιήσουμε το Δέντρο αποφάσεων:

```
from sklearn import tree
decisionTree = tree.DecisionTreeClassifier()
decisionTree.fit(X_train, y_train)
# Και η βαθμολογία που λαμβάνουμε:
decisionTree.score(X_test, y_test)
```

και παίρνουμε σαν έξοδο:

```
= RESTART: C:/Users/
0.5918367346938775
|
```

59%! Έστω για λίγο, έχουμε έναν νικητή

Τελικά καταφέραμε να πάρουμε μια καλύτερη βαθμολογία!

Όπως βλέπουμε, με συνεχείς δοκιμές, βρίσκουμε τελικά το μοντέλο το οποίο μπορούμε να ακολουθούμε για να κάνουμε τις αναλύσεις μας.

Κάτι τελευταίο σχετικά με το Decision Tree: μπορούμε εύκολα να εμφανίσουμε το δέντρο χάρη στη βιβλιοθήκη GraphViz.

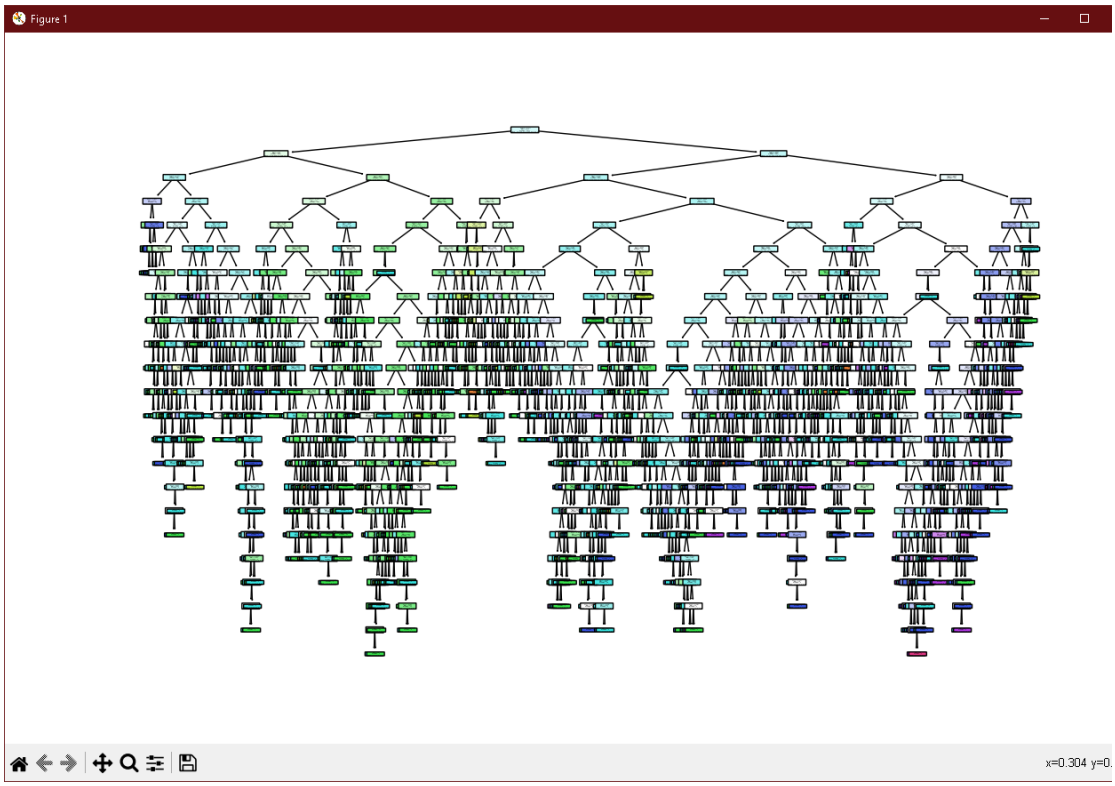
Αυτό το δέντρο είναι απαραίτητο για να κατανοήσουμε πώς κάνει τις επιλογές του ο Αλγόριθμος Μηχανικής Μάθησης. Αυτό είναι ένα από τα χαρακτηριστικά που το καθιστά απαραίτητο για τους Επιστήμονες Δεδομένων και τους Μηχανικούς Μηχανικής Μάθησης!

Δίνοντας ολόκληρο τον κώδικά μας έχουμε:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

# Φόρτωση του dataset
df = pd.read_csv("winequality-white.csv", sep=";")

# Προετοιμασία χαρακτηριστικών κι ετικετών
df_features = df.drop(columns='quality')
df_label = df['quality']

# Διαχωρισμός δεδομένων για εκπαιδευση και τεστ
X_train, X_test, y_train, y_test =
train_test_split(df_features, df_label, test_size=0.20)

# Επαυξηση των δεδομένων με το StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Δημιουργία ταξινοξητικού δένδρου αποφάσεων
decisionTree = DecisionTreeClassifier()
decisionTree.fit(X_train, y_train)
```

```
# Γράφημα δένδρου
plt.figure(figsize=(16, 8))

# Εμφάνιση του γραφήματος με διασαφήνιση των παραμέτρων
plot_tree(decisionTree,
          filled=True,
          rounded=True)

# Display the plot
plt.show()
```

Ας δούμε το γράφημα:



Εδώ τελειώνουμε την εισαγωγή μας στη μηχανική μάθηση και ολοκληρώνουμε με 3 απλές ασκήσεις

28.2.0 Ασκήσεις

Άσκηση 1: Data Splitting

Διαχωρίστε το dataset των κρασιών σε δύο training και testing sets χρησιμοποιώντας μια ποσοστοποίηση 70% - 30%.

Άσκηση 2: Εκπαίδευση μοντέλου

Εκπαιδεύστε ένα μοντέλο Logistic Regression επάνω στο wine dataset χρησιμοποιώντας τα training data και μετά, τυπώστε την ακρίβεια του μοντέλου επάνω στα testing data.

Άσκηση 3: Οπτικοποίηση δένδρου αποφάσεων

Εκπαιδεύστε έναν ταξινομητή δένδρου αποφάσεων επάνω στο wine dataset χρησιμοποιώντας τα παραπάνω training data, και μετά οπτικοποιήστε το δένδρο αποφάσεων.