Module 2

This section focuses on retrieving data from various sources, emphasizing practical techniques and considerations.

Retrieving Data from CSV Files

- CSV (Comma Separated Values) files consist of rows of data separated by commas, easily read using Pandas with a few lines of code.
- Key arguments for the `pd.read_csv` function include specifying delimiters, handling headers, and defining null values.

Retrieving Data from JSON Files

- JSON (JavaScript Object Notation) files are commonly used for data storage and resemble Python dictionaries.
- To read JSON files, the `pd.read_json` function is used, and understanding the structure of the JSON is crucial for successful data retrieval.

```
# Different delimiters - tab-separated file (.tsv):
data = pd.read_csv(filepath, sep='\t')

# Different delimiters - space-separated file:
data = pd.read_csv(filepath, delim_whitespace=True)

# Don't use first row for column names:
data = pd.read_csv(filepath, header=None)

# Specify column names:
data = pd.read_csv(filepath, names=['Name1', 'Name2'])

# Custom missing values:
data = pd.read_csv(filepath, na_values=['NA', 99])
```

Working with SQL and NoSQL databases, as well as accessing data through APIs and cloud sources.

SQL Databases

- SQL (Structured Query Language) is used for relational databases with a fixed schema.
- Examples include Microsoft SQL Server, Postgres, MySQL, and Oracle DB. Python libraries like sqlite3 and SQLAlchemy facilitate database connections.

Read SQL Databases

```
# SQL Data Imports
import sqlite3 as sq3
import pandas as pd

# Initialize path to SQLite database
path = 'data/classic_rock.db'

# Create connection SQL database
con = sq3.Connection(path)

# Write query
query = ''' SELECT * FROM rock_songs;
'''
# Execute query
data = pd.read_sql(query, con)
```

NoSQL Databases

- NoSQL databases are non-relational and can vary in structure, often storing data in JSON format.
- Types include document databases (like MongoDB), graph databases (for network analysis), and wide column families.

Examples of NoSQL databases:

- Document databases: mongoDB, couchDB

- Key-value stores: Riak, Voldemort, Redis
- Graph databases: Neo4j, HyperGraph
- Wide-column stores: Cassandra, HBase

Read NoSQL Databases

```
# SQL Data Imports
from pymongo import MongoClient

# Create a Mongo connection
con = MongoClient()

# Choose database (con.list_database_names()
# will display available databases)
db = con.database_name

# Create a cursor object using a query
cursor = db.collection_name.find(query)

# Expand cursor and construct DataFrame
df = pd.DataFrame(list(cursor))
```

Data Access via APIs and Cloud

- APIs allow access to data from various providers, such as Twitter or Amazon.
- Datasets can also be accessed online, for example, using Pandas to read CSV files from URLs.

```
#UCI Cars data set - url location
data_url =
'http://archive.ics.uci.edu/ml/machine
-learning-databases/autos/imports-
85.data'

#Read data into Pandas
df=pd.read_csv(data_url, header=None)
```

The content emphasizes the importance of understanding different data sources and how to retrieve and manipulate data effectively.

Significance of data cleaning in the machine learning process.

Importance of Data Cleaning

Decisions and analytics are increasingly driven by data and models.

Key aspects of Machine Learning Workflow depend on cleaned data:

Observations: An instance of the data (usually a point or row in a dataset)

- Labels: Output variables) being predicted
- Algorithms: Computer programs that estimate models based on available data
- Features: Information we have for each observation (variables)
- Model: Hypothesized relationship between observations and data

Common Issues with Messy Data

- Companies face challenges like lack of relevant data, excessive data spread across environments, and poor data quality.
- Messy data can include duplicates, inconsistent text, missing values, and outliers, which can skew results and complicate analysis.

Handling Duplicates and Data Quality

- Identifying and managing duplicate data is essential; some duplicates may be valid while others may introduce noise.
- It's important to filter data carefully to maintain access to useful information while addressing duplicates and ensuring data quality.

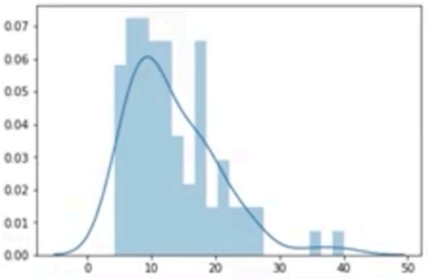
Handling Missing Values

- **Removing Data**: You can remove entire rows to clean the dataset quickly, but this may lead to loss of important information if many rows are missing values.
- **Imputation**: Replacing null values with the mean or median helps retain data but introduces uncertainty as these are estimates.
- **Masking**: Treating missing values as a separate category can provide insights, but it also adds uncertainty by assuming all missing values are similar.

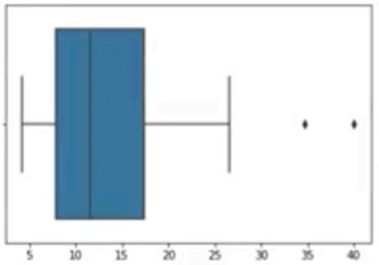
Dealing with Outliers

- **Definition**: Outliers are data points that significantly differ from others and can skew model predictions.
- **Identification**: Use visualizations like histograms, density plots, and box plots to detect outliers.
- **Analysis**: While some outliers may be erroneous, they can also provide valuable insights into the data, so they should be investigated rather than simply removed.

How to Find Outliers?



Plots
Histogram, Density Plot, Box Plot



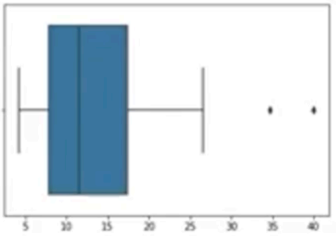
Statistics
Interquartile Range, Standard Deviation

```
import numpy as np

# calculate the interquartile range
q25, q50, q75 = np.percentile(data, [25, 50, 75])
iqr = q75 - q25

# calculate the min / max limits to be considered an outlier
min = q25 - 1.5*(iqr)
max = q75 + 1.5*(iqr)

print(min, q25, q50, q75, max)
```

```
-6.6 7.8 11.5 17.4 31.8
```

```
# identify the points
[x for x in data['Unemployment'] if x > max]
```

```
[40.0, 34.700000000000003]
```



Interquartile Range

Residuals and Their Importance

- Residuals are the differences between actual values and predicted values from a model, indicating model failure.
- Standardized residuals help account for varying outcome ranges by dividing residuals by the standard error.

Methods for Handling Residuals

- Deleted residuals involve removing an observation from the dataset to see how it affects model predictions.
- Studentized residuals standardize deleted residuals to assess their impact on the model.

Dealing with Outliers

- Outliers can be removed, but this may lead to loss of important data.

- Assigning a different value to outliers or transforming the data can mitigate their effects.
- Predicting outlier values using similar observations or regression can also be effective, though it may require more effort.

Overall, the discussion emphasizes the significance of data cleaning and the various strategies for managing outliers to ensure robust model performance.