Sudhanshu Agarwal SI 618 Cheatsheet Homework 8

Pandas

Pandas allows us to analyze big data and make conclusions based on statistical theories.
Pandas can clean messy data sets, and make them readable and relevant.

Data Structures:
Series: 1D labeled array.
DataFrame: 2D labeled data structure.

Essential Operations:
1. Read Data: CSV, Excel.
2. Display Data: head(), tail().
3. Indexing: df['col'], loc[], iloc[].
4. Manipulation: Filter, Sort, Group, Handle Missing Data.
5. Cleaning: Drop, Rename.
6. Visualization: Basic plots.
7. Export Data: CSV, Excel.

Correlation:
A statistical measure that describes the extent to which two variable
Range:

Correlation coefficients range from -1 to 1.
- -1: Perfect negative correlation.
- 0: No correlation.
- 1: Perfect positive correlation.
-

Types of Correlation:
- Positive Correlation: Both variables move in the same direct
- Negative Correlation: Variables move in opposite directions.

Strength of Correlation:
- Closer to -1 or 1 indicates a stronger correlation.
- Closer to 0 indicates a weaker correlation.

correlation_matrix = df.corr()

Day 7: |
SPACY: Spacy is an open-source natural language processing (NLP) library for Python. It is designed to be efficient, fast, and easy to use for various NLP tasks.
Key features of Spacy include:
- Tokenization: Spacy can break down a text into individual words or tokens.
```
tokens = nlp(text)
for token in tokens:
   print(token.text)
```
- Part-of-speech (POS) tagging: It can assign grammatical parts of speech (e.g., noun, verb, adjective) to each token.
```
tokens = nlp(text)
for token in tokens:
   print(f"{token.text}: {token.pos_}")
```
- Named Entity Recognition (NER): Spacy can identify and classify named entities in text, such as names of people, organizations, locations, etc.
```
doc = nlp(text)
for ent in doc.ents:
   print(f"{ent.text}: {ent.label_}")
```

Stop Words
Stop words are commonly used words that are typically filtered out or ignored during the processing of natural language text because they are considered to be of little value in helping to analyze the content of a document.
Examples of common stop words in English include "the," "and," "is," "in," "to," "of," et

Word embedding: Type of representation of words in a vector space, where words with similar meanings are mapped to similar vectors.
```
word = "example"
embedding = nlp(word).vector
```
Word vectors, often obtained through word embeddings, can be used to perform interesting semantic operations, such as analogies. One common example of an analogy in the context of word vectors is the well-known "king - man + woman = queen" analogy. This demonstrates that word vectors can capture relationships between words and perform operations like addition and subtraction.

Sentiment Analysis with NLTK
Sentiment analysis is the process of determining the sentiment or opinion expressed in a piece of text. The Natural Language Toolkit (NLTK) is a powerful library in Python for natural language processing, and it includes tools for sentiment analysis.

Numpy

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Essential Operations:

1. Array Creation:
- numpy.array(): From lists or tuples.
- numpy.zeros(), numpy.ones(): Zeros or ones arrays.
1. Array Operations:
- +, -, *, /: Element-wise arithmetic.
- numpy.dot(): Matrix multiplication.
- numpy.transpose(): Transpose array.
- numpy.sum(), numpy.mean(), numpy.max(), numpy.min(): Aggregates.
1. Indexing and Slicing:
- array[index], array[start:stop], array[:, 1]: Access and slice.
1. Array Manipulation:
- numpy.reshape(), numpy.concatenate(): Reshape and concatenate.
- numpy.split(), numpy.append(), numpy.insert(), numpy.delete(): Modify.
1. Mathematical Functions:
- numpy.sqrt(), numpy.power(), numpy.abs(): Square root, power, absolute.
1. Statistical Functions:
- numpy.mean(), numpy.median(), numpy.std(), numpy.var(): Mean, median, std, var.

Regression:
A statistical method that models the relationship betwee
equation to observed data.

$$Y = b_0 + b_1 X + \varepsilon$$

Where:
Y: Dependent variable
X: Independent variable
b_0: $Y-$ intercept.
b_1: Slope of the line
ε: Error term.

Residuals: The difference between the observed and predicted values.
Goodness of Fit: R-squared measures the proportion of the variance in the dependent variable explained by the independent variable(s).

OLS:
OLS is a method used in linear regression to estimate the parameters of a linear model by minimizing the sum of squared differences between the observed and predicted values.
model = smf.ols(formula=formula, data=df).fit()

Assumptions:
Linearity: The relationship between independent and dependent variables is linear.
Independence: Residuals are independent.
Homoscedasticity: Residuals have constant variance.
Normality: Residuals are normally distributed.
No perfect multicollinearity: Independent variables are not perfectly correlated.

Diagnostics and Metrics:
R-squared: Proportion of variance explained by the model.
Residuals: Differences between observed and predicted values.
Coefficient of Determination (R-squared): Measures the goodness of fit.
Adjusted (R-squared): Accounts for the number of predictors in the model.

Potential Pitfalls:
Overfitting: Including too many variables can lead to overfitting.
Multicollinearity: High correlation among independent variables can cause issues.
Heteroscedasticity: Unequal variance of residuals can violate assumptions.

Seaborn:
Seaborn is a Python data visualization library based on Matplotlib that provides a high-level interface for creating informative and aesthetically pleasing statistical graphics. It simplifies the process of generating common statistical plots, enhances visual appeal, and facilitates exploratory data analysis by offering a concise and intuitive API.

Scatter Plot: Displays the relationship between two numerical variables through individual data points.
sns.scatterplot(x='x_var', y='y_var', data=df)
Line Plot: Illustrates the trend or pattern in data points connected by straight lines.
sns.lineplot(x='x_var', y='y_var', data=df)
Bar Plot: Represents categorical data with rectangular bars, emphasizing comparisons between different groups.
sns.barplot(x='x_var', y='y_var', data=df)
Box Plot: Summarizes the distribution of numerical data, highlighting median, quartiles, and potential outliers.
sns.boxplot(x='x_var', y='y_var', data=df)
Violin Plot: Combines aspects of box plots and kernel density plots to show the distribution of data across different categories.
sns.violinplot(x='x_var', y='y_var', data=df)
Pair Plot: Presents pairwise relationships between numerical variables in a matrix of scatterplots.
sns.pairplot(df)
Heatmap: Visualizes the correlation or magnitude of values in a 2D dataset using colors.
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
Facet Grid: Creates a grid of subplots based on categorical variables, allowing the comparison of multiple plots.
g = sns.FacetGrid(df, col='category', hue='group')
g.map(sns.scatterplot, 'x_var', 'y_var')
Regression Plot: Shows the relationship between two variables along with a fitted regression line.
sns.regplot(x='x_var', y='y_var', data=df)

Hypothesis Testing Overview:
Hypothesis testing is a statistical method used to make inferences about population parameters based on a sample of data. The process involves formulating null and alternative hypotheses and using statistical tests to determine if there's enough evidence to reject the null hypothesis.

Formulate Hypotheses:
- Null Hypothesis: Assumes no effect or no difference.
- Alternative Hypothesis: Posits an effect or difference.

Choose Significance Level (α):
- Common values are 0.05, 0.01, or 0.10.
- Represents the probability of rejecting a true null hypothesis (Type I error).

Select Appropriate Test:
- t-test: Compares means of two groups.
- ANOVA (Analysis of Variance): Compares means of more than two groups.

Matplotlib:
Matplotlib is a Python library for creating static, animated, and interactive visualizations. It supports a wide range of plot types and provides extensive customization options. Matplotlib is widely used in data analysis, scientific research, and visualization tasks.

Line Plot:
  plt.plot(x, y): Plotting a line.
  plt.scatter(x, y): Scatter plot.
Bar Plot:
  plt.bar(x, y): Bar chart.
  plt.barh(x, y): Horizontal bar chart.
Pie Chart:
  plt.pie(sizes, labels=labels): Pie chart.
Histogram:
  plt.hist(data, bins=n): Histogram.

Customization:
Labels and Title:
  plt.xlabel('x-axis label'), plt.ylabel('y-axis label'): Axis labels.
  plt.title('Title'): Plot title.
Legends and Annotations:
  plt.legend(): Show legend.

Subplots:
Creating Subplots:
  plt.subplot(rows, cols, index): Divide plot into subplots.

Multi-Plot Layouts:
  plt.subplots(rows, cols): Create subplots grid.

Day 6:
Pivot table: The pivot_table function is specifically designed for reshaping and summarizing data in a tabular form.
pivot_df = pd.pivot_table(df, values='Value', index='Date', columns='Category', aggfunc='sum')
Cross Tab: The pd.crosstab function in pandas is another way to create a cross-tabulation (cross-tab) of two or more factors.
pandas.crosstab(index, columns, values=None, rownames=None, colnames=None, aggfunc=None)
Regular Expression: Regular expressions (regex or regexp) are powerful tools for pattern matching and manipulation of strings. They provide a concise and flexible means for searching, matching, and manipulating text. In Python, the re module is used to work with regular expressions.
```
import re
pattern = r"your_regular_expression_here"
```

Common Patterns:
\d: Matches any digit.
\w: Matches any alphanumeric character (word character).
\s: Matches any whitespace character.
.: Matches any character except a newline.
^: Anchors the regex at the start of the string.
$: Anchors the regex at the end of the string.

Quantifiers:
*: Matches 0 or more occurrences.
+: Matches 1 or more occurrences.
?: Matches 0 or 1 occurrence.
{n}: Matches exactly n occurrences.
{n,}: Matches n or more occurrences.
{n,m}: Matches between n and m occurrences.

Chi-Squared Test Types:
Objective: Determine if there is a significant association between two categorical variables.
Hypotheses:
- Null Hypothesis: No association exists between the variables.
- Alternative Hypothesis There is an association between the variables.

Test Statistic:
Chi-squared statistic is calculated based on the difference between observed and expected frequencies.
Degrees of Freedom: For a contingency table with r rows and c columns, degrees of freedom = (r - 1) * (c - 1).
chi2, p = stats.chisquare(f_obs=observed, f_exp=expected)

Features: Features are the input variables or attributes used by a machine learning model to make predictions or classifications.

Types of Features:

- Numerical Features: Examples: Age, Income, Temperature
- Continuous values that can be measured on a numeric scale.
- Categorical Features: Examples: Gender, Color, Country
- Represent categories and often require encoding.

Feature Engineering:
The process of selecting, transforming, or creating new features to enhance model performance. Techniques include normalization, scaling, one-hot encoding, and creating interaction terms.
Feature Selection: Choosing the most relevant features to improve model efficiency and generalization. Methods include statistical tests, feature importance from models, and recursive feature elimination.

Labels: The label, also known as the target or output, is the variable the machine learning model is designed to predict or classify.
Types of Labels:
Binary Labels: Two possible outcomes (0 or 1, True or False).
Example: Spam or Not Spam.
Multiclass Labels: More than two categories.
Example: Animal classification (Cat, Dog, Bird).
Regression Labels: Continuous numerical values.
Example: Predicting house prices.

Evaluation Metrics:

- Binary Classification: Accuracy, Precision, Recall, F1 Score, ROC-AUC.
- Multiclass Classification: Accuracy, Precision, Recall, F1 Score, Confusion Matrix.
- Regression: Mean Absolute Error (MAE), Mean Squared Error (MSE), R-squared.

Handling Missing Labels:
Strategies include removing instances with missing labels, imputing missing labels, or using special models for handling missing data.

Train-Test Split:
Objective: Assess the model's performance on unseen data.
Process: Dividing the dataset into two subsets: training set and testing set.
Split Ratio: Typical Split: 70-30 or 80-20 for larger datasets.
Considerations: Adjust based on dataset size and modeling goals.
Implementation:

- Random Splitting: Use random sampling to ensure a representative distribution in both sets.
- Stratified Splitting: Maintain the same class distribution in both sets, critical for imbalanced datasets.

Supervised Learning:
Task: Predict or classify based on labeled training data.
Input-Output Mapping: Learn the mapping from input features to corresponding labeled outputs.
Data:
Labeled Data: Training dataset includes input features along with corresponding correct output labels.
Examples: Classification (e.g., spam detection), Regression (e.g., house price prediction).

Algorithm Types: Linear Regression, Support Vector Machines (SVM), Decision Trees, Neural Networks.
Metrics: Accuracy, Precision, Recall, F1 Score (classification), Mean Squared Error, R-squared (regression).

Unsupervised Learning:
Task: Extract patterns, relationships, or structure from unlabeled data.
Input-Output Mapping: No predefined output labels; the algorithm discovers hidden patterns or groups.
Unlabeled Data: No explicit output labels in the training dataset.
Examples: Clustering (e.g., customer segmentation), Dimensionality Reduction (e.g., PCA), Association (e.g., market basket analysis).

Algorithm Types: K-Means Clustering, Principal Component Analysis (PCA), Apriori Algorithm.
Evaluation is more subjective: Metrics depend on the specific task.
Clustering: Silhouette Score, Davies-Bouldin Index.

What is a Machine Learning Pipeline?
A machine learning pipeline is a systematic way to organize and streamline the machine learning workflow. I

```
eg - numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
```

Regression in Machine Learning:
Regression is a type of supervised learning where the goal is to predict a continuous target variable based on one or more input features. It is commonly used for tasks such as predicting house prices, stock prices, or any numerical outcome.

Steps in Building a Regression Model:
Data Exploration, Data Preprocessing, Feature Selection/Engineering, Train-Test Split, Model Selection, Model Training, Model Evaluation, Fine-tuning.
Machine Learning Pipeline with Regression:
Create Pipeline, Hyperparameter Tuning, Scalability, Deployment.

One-Hot Encoding:
Convert categorical variables into a binary matrix format to make them suitable for machine learning algorithms that require numerical input.

```
encoder = OneHotEncoder()
encoded_features = encoder.fit_transform(categorical_feature.reshape(-1, 1)).toarray()
```

Standard Scaling (Z-score normalization):
Standardize numerical features to have zero mean and unit variance.
Useful when features have different scales, ensuring that each feature contributes equally to the model.

```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(numerical_feature.reshape(-1, 1))
```

Dimension Reduction:

- Reduce the number of features while preserving the essential information in the data.
- Mitigate the curse of dimensionality and improve computational efficiency.

Principal Component Analysis (PCA):

- Linear dimensionality reduction technique that identifies the most important features (principal components) in the data.

Standardize Data: Ensure all features have a mean of 0 and a standard deviation of 1.
Covariance Matrix: Calculate the covariance matrix of the standardized features.
Eigendecomposition: Decompose the covariance matrix into eigenvectors and eigenvalues.
Select Components: Choose the top-k eigenvectors corresponding to the largest eigenvalues to form the new feature space.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

t-Distributed Stochastic Neighbor Embedding (t-SNE): Non-linear dimensionality reduction technique that focuses on preserving local similarities between data points.

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, perplexity=30)
X_tsne = tsne.fit_transform(X)
```

Choosing Between PCA and t-SNE:
Use PCA When:

- Linearity Assumption Holds: The underlying relationships in the data are primarily linear.
- Computational Efficiency: Faster for large datasets and high-dimensional spaces.

Use t-SNE When:

- Non-Linearity is Present: Data has complex, non-linear relationships that PCA may not capture well.
- Visualization: Especially useful for visualizing clusters and local relationships in lower-dimensional space.

Classification:
Assign predefined labels (classes) to input data points based on their features.
Selecting a Classifier:
Nature of Data: Linear vs. Non-linear relationships, separability.
Data Size: Large datasets may benefit from scalable algorithms.
Interpretability: Decision trees and linear models are often more interpretable.
Computational Resources: Choose models based on computational efficiency.
Example: Decision trees

```
from sklearn.tree import DecisionTreeClassifier
tree_classifier = DecisionTreeClassifier(max_depth=3, random_state=42)
tree_classifier.fit(X_train, y_train)
y_pred = tree_classifier.predict(X_test)
```

Simple Classifiers:
Logistic Regression:
Linear model for binary and multiclass classification.

```
from sklearn.linear_model import LogisticRegression
logistic_classifier = LogisticRegression()
```

k-Nearest Neighbors (k-NN):
Classifies based on the majority class among the k-nearest neighbors.

```
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier(n_neighbors=3)
```

Docker Basics:

- Docker is a platform for developing, shipping, and running applications in containers.
- Container: A lightweight, portable, and self-sufficient unit that can run applications and their dependencies isolated from the host system.
- Docker Image: A snapshot of a file system and a set of parameters, used to create and run containers.
- Docker Container: An instance of a Docker image, running as a process on the host machine.

What is Dask?

Dask is a parallel computing library in Python that enables scalable, flexible, and efficient parallel computing.
Key Features:

- Parallelization: Enables parallel computing for handling larger-than-memory datasets.
- Dynamic Task Scheduling: Optimizes task execution by dynamically scheduling tasks on available resources.
- Familiar API: Dask provides APIs similar to NumPy, pandas, and scikit-learn for seamless integration into existing workflows.

What is Multithreading?
Multithreading is a concurrent execution model where multiple threads within a process share the same data space, allowing for parallel execution of tasks. In Python, the threading module is commonly used for implementing multithreading.

Spark:

- Apache Spark is an open-source, distributed computing system that provides a fast and general-purpose cluster-computing framework for big data processing. It's designed to be fast, flexible, and easy to use.
- In-Memory Processing: Spark stores intermediate data in-memory, reducing the need for frequent disk I/O and enhancing performance.
- Distributed Computing: Spark distributes data and computation across a cluster of machines, enabling parallel processing for large-scale data.
- Resilient Distributed Datasets (RDDs): Fundamental data structure in Spark that allows fault-tolerant parallel processing of data.
- Spark SQL: Module for working with structured and semi-structured data, providing a DataFrame API and support for SQL queries.

Big Data Processing with Spark:

- Data Processing: Spark is well-suited for processing large volumes of data, transforming and aggregating it efficiently.
- Batch Processing: Spark supports batch processing for processing data in large, static datasets.
- Iterative Algorithms: Spark's in-memory processing is beneficial for iterative algorithms common in machine learning and graph processing.
- Stream Processing: Spark Streaming allows the processing of real-time data streams.

Advantages:

- Speed: Spark's in-memory processing significantly speeds up data processing compared to traditional batch processing frameworks.
- Ease of Use: Provides high-level APIs in Java, Scala, Python, and R, making it accessible to a broader audience.
- Flexibility: Supports various data processing tasks, from batch processing to real-time stream processing.
- Unified Platform: Offers a unified platform for big data processing, eliminating the need for separate tools for different tasks.