Utilizing Pandas AI for Data Analysis

Bring the latest AI implementation to Pandas to improve your data workflow.

By **Cornellius Yudha Wijaya**, KDnuggets Technical Content Specialist on April 16, 2024 in **Data Science**

Are you proficient in the data field using Python? If so, I bet most of you use Pandas for data manipulation.

If you don't know, Pandas is an open-source Python package specifically developed for data analysis and manipulation. It's one of the most-used packages and one you usually learn when starting a data science journey in Python.

So, what is Pandas AI? I guess you are reading this article because you want to know about it.

Well, as you know, we are in a time when Generative AI is everywhere. Imagine if you can perform data analysis on your data using Generative AI; things would be much easier.

This is what Pandas AI brings. With simple prompts, we can quickly analyze and manipulate our dataset without sending our data somewhere.

This article will explore how to utilize Pandas AI for Data Analysis tasks. In the article, we will learn the following:

- Pandas AI Setup
- Data Exploration with Pandas AI
- Data Visualization with Pandas AI
- Pandas AI Advanced usage

If you are ready to learn, let's get into it!

Pandas AI Setup

Pandas AI is a Python package that implements a Large Language Model (LLM) capability into Pandas API. We can use standard Pandas API with Generative AI enhancement that turns Pandas into a conversational tool.

We mainly want to use Pandas AI because of the simple process that the package provides. The package could automatically analyze data using a simple prompt without requiring complex code.

Enough introduction. Let's get into the hands-on.

First, we need to install the package before anything else.

```
pip install pandasai
```

Next, we must set up the LLM we want to use for Pandas AI. There are several options, such as OpenAI GPT and HuggingFace. However, we will use the OpenAI GPT for this tutorial.

Setting the OpenAI model into Pandas AI is straightforward, but you would need the OpenAI API Key. If you don't have one, you can get on their website.

If everything is ready, let's set up the Pandas AI LLM using the code below.

```
from pandasai.llm import OpenAI

llm = OpenAI(api_token="Your OpenAI API Key")
```

You are now ready to do Data Analysis with Pandas AI.

Data Exploration with Pandas AI

Let's start with a sample dataset and try the data exploration with Pandas AI. I would use the Titanic data from the Seaborn package in this example.

```
import seaborn as sns
```


7 Best Platforms to Practice Pythor
5 Free Advanced Python Programm Courses
A Starter Guide to Data Structures and Machine Learning
Free Google Cloud Learning Path fc Gemini
7 End-to-End MLOps Platforms You Must Try in 2024
5 Free Stanford University Courses Learn Data Science
Data Scientist Breakdown: Skills, Certifications, and Salary
Free Python Resources That Can H You Become a Pro
Integrating Generative AI in Conter Creation

After that, we can perform conversational activity on our DataFrame.

Let's try a simple question.

```
response = df.chat("""Return the survived class in percentage""")

response
```

The percentage of passengers who survived is: 38.38%

From the prompt, Pandas AI could come up with the solution and answer our questions.

We can ask Pandas AI questions that provide answers in the DataFrame object. For example, here are several prompts for analyzing the data.

```
#Data Summary
summary = df.chat("""Can you get me the statistical summary of the dataset""")

#Class percentage
surv_pclass_perc = df.chat("""Return the survived in percentage breakdown by p

#Missing Data
missing_data_perc = df.chat("""Return the missing data percentage for the colu

#Outlier Data
outlier_fare_data = response = df.chat("""Please provide me the data rows that
contains outlier data based on fare column""")
```

df.chat("""Can you get me the statistical summary of the dataset""")

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

df.chat("""Return the survived in percentage breakdown by pclass""")

	pclass	survived_percentage
0	1	62.962963
1	2	47.282609
2	3	24.236253

df.chat("""Return the missing data percentage for the columns""")

	missing_data_percentage
survived	0.000000
pclass	0.000000
sex	0.000000
age	19.865320
sibsp	0.000000
parch	0.000000
fare	0.000000
embarked	0.224467
class	0.000000
who	0.000000
adult_male	0.000000
deck	77.216611
embark_town	0.224467
alive	0.000000
alone	0.000000

df.chat("""Please provide me the data rows that contains outlier data based on fare column""")

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
258	1	1	female	35.0	0	0	512.3292	C	First	woman	False	NaN	Cherbourg	yes	True
779	1	1	female	43.0	0	1	211.3375	S	First	woman	False	B	Southampton	yes	False
527	0	1	male	NaN	0	0	221.7792	S	First	man	True	C	Southampton	no	True
27	0	1	male	19.0	3	2	263.0000	S	First	man	True	C	Southampton	no	False
679	1	1	male	36.0	0	1	512.3292	C	First	man	True	B	Cherbourg	yes	False
299	1	1	female	50.0	0	1	247.5208	C	First	woman	False	B	Cherbourg	yes	False
557	0	1	male	NaN	0	0	227.5250	C	First	man	True	NaN	Cherbourg	no	True
689	1	1	female	15.0	0	1	211.3375	S	First	child	False	B	Southampton	yes	False
438	0	1	male	64.0	1	4	263.0000	S	First	man	True	C	Southampton	no	False
311	1	1	female	18.0	2	2	262.3750	C	First	woman	False	B	Cherbourg	yes	False
700	1	1	female	18.0	1	0	227.5250	C	First	woman	False	C	Cherbourg	yes	False
716	1	1	female	38.0	0	0	227.5250	C	First	woman	False	C	Cherbourg	yes	True
341	1	1	female	24.0	3	2	263.0000	S	First	woman	False	C	Southampton	yes	False
88	1	1	female	23.0	3	2	263.0000	S	First	woman	False	C	Southampton	yes	False
730	1	1	female	29.0	0	0	211.3375	S	First	woman	False	B	Southampton	yes	True
737	1	1	male	35.0	0	0	512.3292	C	First	man	True	B	Cherbourg	yes	True
742	1	1	female	21.0	2	2	262.3750	C	First	woman	False	B	Cherbourg	yes	False
118	0	1	male	24.0	0	1	247.5208	C	First	man	True	B	Cherbourg	no	False
377	0	1	male	27.0	0	2	211.5000	C	First	man	True	C	Cherbourg	no	False
380	1	1	female	42.0	0	0	227.5250	C	First	woman	False	NaN	Cherbourg	yes	True

Image by Author

You can see from the image above that the Pandas AI can provide information with the DataFrame object, even if the prompt is quite complex.

However, Pandas AI can't handle a calculation that is too complex as the packages are limited to the LLM we pass on the SmartDataFrame object. In the future, I am sure that Pandas AI could handle much more detailed analysis as the LLM capability is evolving.

specify the prompt, Pandas AI will give the visualization output.

Let's try a simple example.

```
response = df.chat('Please provide me the fare data distribution visualization

response
```

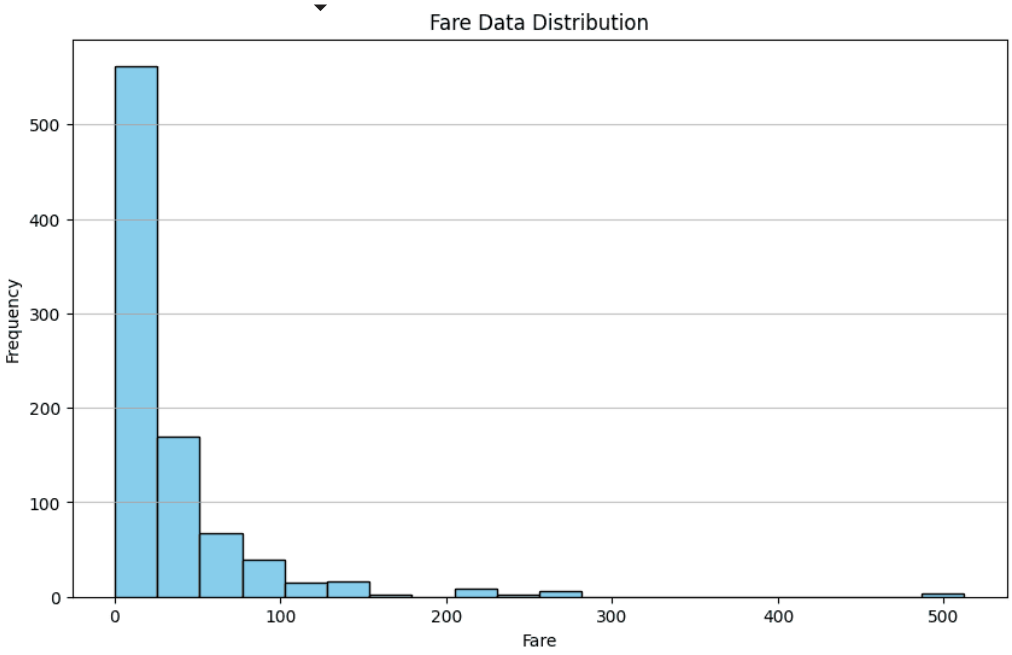

Image by Author

In the example above, we ask Pandas AI to visualize the distribution of the Fare column. The output is the Bar Chart distribution from the dataset.

Just like Data Exploration, you can perform any kind of data visualization. However, Pandas AI still can't handle more complex visualization processes.

Here are some other examples of Data Visualization with Pandas AI.

```
kde_plot = df.chat("""Please plot the kde distribution of age column and separ

box_plot = df.chat("""Return me the box plot visualization of the age column s

heat_map = df.chat("""Give me heat map plot to visualize the numerical columns

count_plot = df.chat("""Visualize the categorical column sex and survived""")
```

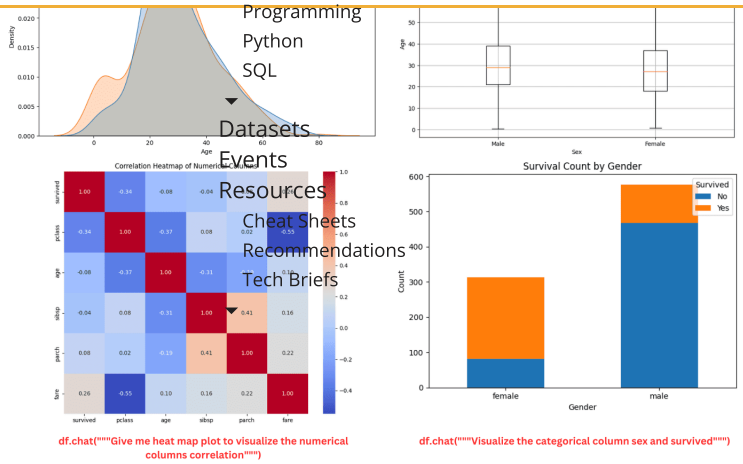

Image by Author

The plot looks nice and neat. You can keep asking the Pandas AI for more details if necessary.

Pandas AI Advances Usage

We can use several in-built APIs from Pandas AI to improve the Pandas AI experience.

Cache clearing

By default, all the prompts and results from the Pandas AI object are stored in the local directory to reduce the processing time and cut the time the Pandas AI needs to call the model.

However, this cache could sometimes make the Pandas AI result irrelevant as they consider the past result. That's why it's good practice to clear the cache. You can clear them with the following code.

In this way, no prompt or result is stored from the beginning.

Custom Head

It's possible to pass a sample head DataFrame to Pandas AI. It's helpful if you don't want to share some private data with the LLM or just want to provide an example to Pandas AI.

To do that, you can use the following code.

```
from pandasai import SmartDataframe
import pandas as pd

# head df
head_df = data.sample(5)

df = SmartDataframe(data, config={
    "custom_head": head_df,
    'llm': llm
})
```

Pandas AI Skills and Agents

```
from pandasai import Agent
from pandasai.skills import skill

employees_data = {
    "EmployeeID": [1, 2, 3, 4, 5],
    "Name": ["John", "Emma", "Liam", "Olivia", "William"],
    "Department": ["HR", "Sales", "IT", "Marketing", "Finance"],
}

salaries_data = {
    "EmployeeID": [1, 2, 3, 4, 5],
    "Salary": [5000, 6000, 4500, 7000, 5500],
}

employees_df = pd.DataFrame(employees_data)
salaries_df = pd.DataFrame(salaries_data)

# Function doc string to give more context to the model for use of this skill
@skill
def plot_salaries(names: list[str], salaries: list[int]):
    """
    Displays the bar chart  having name on x-axis and salaries on y-axis
    Args:
        names (list[str]): Employees' names
        salaries (list[int]): Salaries
    """
    # plot bars
    import matplotlib.pyplot as plt

    plt.bar(names, salaries)
    plt.xlabel("Employee Name")
    plt.ylabel("Salary")
    plt.title("Employee Salaries")
    plt.xticks(rotation=45)

    # Adding count above for each bar
    for i, salary in enumerate(salaries):
        plt.text(i, salary + 1000, str(salary), ha='center', va='bottom')
    plt.show()


agent = Agent([employees_df, salaries_df], config = {'llm': llm})
agent.add_skills(plot_salaries)

response = agent.chat("Plot the employee salaries against names")
```

The Agent would decide if they should use the function we assigned to the Pandas AI or not.

Combining Skill and Agent gives you a more controllable result for your DataFrame analysis.

We have learned how easy it is to use Pandas AI to help our data analysis work. Using the power of LLM, we can limit the coding portion of the data analysis works and instead focus on the critical works.

In this article, we have learned how to set up Pandas AI, perform data exploration and visualization with Pandas AI, and advance usage. You can do much more with the package, so visit their documentation to learn further.

Cornellius Yudha Wijaya is a data science assistant manager and data writer. While working full-time at Allianz Indonesia, he loves to share Python and data tips via social media and writing media. Cornellius writes on a variety of AI and machine learning topics.

More On This Topic

- Pandas vs. Polars: A Comparative Analysis of Python's Dataframe Libraries
- Introduction to Pandas for Data Science
- Data Ingestion with Pandas: A Beginner Tutorial
- Simplify Data Processing with Pandas Pipeline
- 10 Pandas One Liners for Data Access, Manipulation, and Management
- The Optimal Way to Input Missing Data with Pandas fillna()

<= Previous post

Next post =>