Comprehensive Guide to pandas with the Iris Dataset

Sayan Chaki

March 12, 2025

Contents

1 Introduction

This guide provides a comprehensive overview of using pandas with the Iris dataset. The Iris dataset is a classic dataset in statistics and machine learning, introduced by the British statistician and biologist Ronald Fisher in 1936. It consists of 150 samples from three species of Iris (Iris setosa, Iris virginica, and Iris versicolor), with four features measured for each sample: the length and width of the sepals and petals.

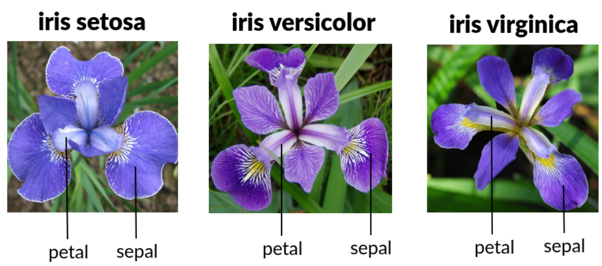

Figure 1: The three species of Iris in the dataset

2 Loading the Dataset

The first step in working with the Iris dataset is to load it into a pandas DataFrame. This can be done in several ways:

2.1 Using scikit-learn

```
from sklearn import datasets
import pandas as pd

# Load the Iris dataset
iris = datasets.load_iris()

# Create a DataFrame
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

# Display the first few rows
print(df.head())
```

Listing 1: Loading Iris dataset from scikit-learn

2.2 Using seaborn

```
import seaborn as sns
import pandas as pd

# Load the Iris dataset
```

```python
iris = sns.load_dataset('iris')

# Display the first few rows
print(iris.head())
```

Listing 2: Loading Iris dataset from seaborn

3 Basic Exploratory Data Analysis

3.1 Dataset Information

Once the dataset is loaded, we can obtain basic information about it:

```python
# Basic information about the dataset
print(df.info())

# Summary statistics
print(df.describe())

# Check for missing values
print(df.isnull().sum())
```

Listing 3: Getting dataset information

The output of `df.info()` would look like:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    category
dtypes: category(1), float64(4)
memory usage: 5.0 KB
```

3.2 Summary Statistics

The output of `df.describe()` can be converted into a LaTeX table:

Table 1: Summary statistics for the Iris dataset

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

3.3 Data Distribution by Species

We can analyze the distribution of samples by species:

```python
# Count of samples by species
species_counts = df['species'].value_counts()
print(species_counts)
```

Listing 4: Distribution by species

This would output:

Table 2: Distribution of samples by species

Species	Count
setosa	50
versicolor	50
virginica	50

4 Data Manipulation Techniques

4.1 Selecting and Filtering Data

```python
# Select specific columns
sepal_data = df[['sepal length (cm)', 'sepal width (cm)']]

# Filter data by species
setosa = df[df['species'] == 'setosa']
versicolor = df[df['species'] == 'versicolor']
virginica = df[df['species'] == 'virginica']

# Display the first few rows of setosa data
print(setosa.head())
```

Listing 5: Selecting and filtering data

4.2 Adding New Columns

```python
# Create a new column for sepal area
df['sepal_area'] = df['sepal length (cm)'] * df['sepal width (cm)']

# Create a new column for petal area
df['petal_area'] = df['petal length (cm)'] * df['petal width (cm)']

# Create a column for sepal length to width ratio
df['sepal_ratio'] = df['sepal length (cm)'] / df['sepal width (cm)']

# Display the first few rows with new columns
print(df.head())
```

Listing 6: Creating new features

4.3 Grouping and Aggregation

```python
# Group by species and calculate mean of each feature
species_means = df.groupby('species').mean()
print(species_means)

# Multiple aggregations
aggs = df.groupby('species').agg({
    'sepal length (cm)': ['min', 'max', 'mean', 'std'],
    'petal length (cm)': ['min', 'max', 'mean', 'std']
})
print(aggs)
```

Listing 7: Grouping and aggregating data

The species means table would look like:

Table 3: Mean values of features by species

species	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	sepal_area	petal_area
setosa	5.006	3.428	1.462	0.246	17.161	0.361
versicolor	5.936	2.770	4.260	1.326	16.444	5.661
virginica	6.588	2.974	5.552	2.026	19.595	11.270

5 Statistical Analysis

5.1 Correlation Analysis

```python
# Calculate correlation matrix
correlation_matrix = df.corr()
print(correlation_matrix)
```

Listing 8: Correlation analysis

The correlation matrix can be presented as:

Table 4: Correlation matrix of Iris features

	sepal length	sepal width	petal length	petal width	sepal_area	petal_area
sepal length	1.000	-0.118	0.872	0.818	0.814	0.876
sepal width	-0.118	1.000	-0.428	-0.366	0.452	-0.417
petal length	0.872	-0.428	1.000	0.963	0.506	0.983
petal width	0.818	-0.366	0.963	1.000	0.501	0.979
sepal_area	0.814	0.452	0.506	0.501	1.000	0.521
petal_area	0.876	-0.417	0.983	0.979	0.521	1.000

5.2 Descriptive Statistics by Group

```
# Calculate descriptive statistics by species
stats_by_species = df.groupby('species').describe()
print(stats_by_species)
```

Listing 9: Descriptive statistics by species

6 Data Visualization with pandas

Here are some common visualization techniques that can be used with pandas and subsequently included in LaTeX documents:

6.1 Histograms

```
import matplotlib.pyplot as plt

# Create histograms for sepal length by species
plt.figure(figsize=(10, 6))
for species in iris.species.unique():
    subset = iris[iris.species == species]
    plt.hist(subset['sepal length (cm)'], alpha=0.5, label=species)
plt.legend()
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')
plt.title('Distribution of Sepal Length by Species')
plt.savefig('sepal_length_hist.png', dpi=300, bbox_inches='tight')
plt.close()
```

Listing 10: Creating histograms

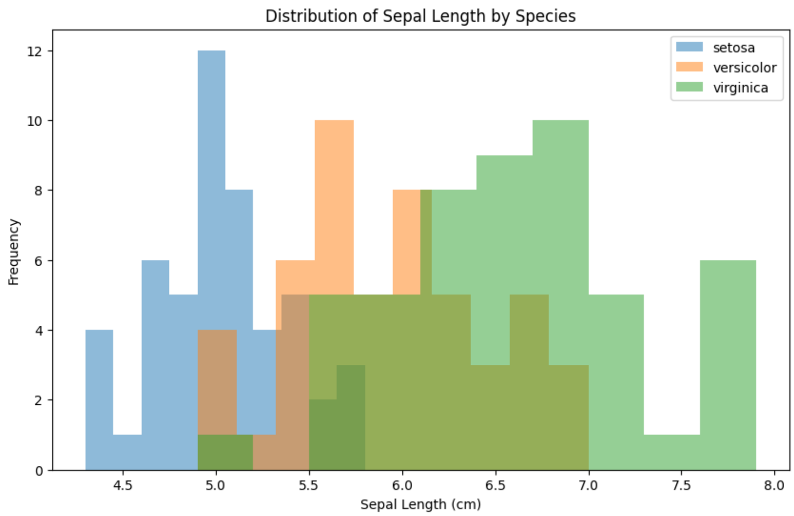

Figure 2: Distribution of sepal length by species

6.2 Box Plots

```
# Create box plots for all features by species
plt.figure(figsize=(12, 8))
df.boxplot(by='species', figsize=(12, 8))
plt.savefig('boxplots.png', dpi=300, bbox_inches='tight')
plt.close()
```

Listing 11: Creating box plots

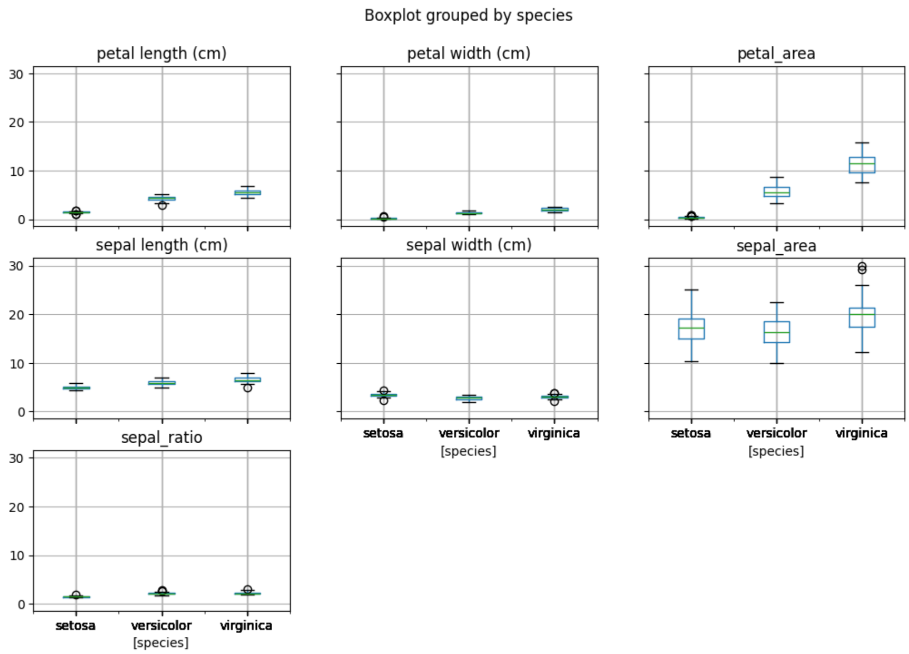

Figure 3: Box plots of Iris features by species

6.3 Scatter Plots

```
# Create scatter plot of sepal length vs width
plt.figure(figsize=(8, 6))
for species, color in zip(iris.species.unique(), ['blue', 'green', 'red']):
    subset = iris[iris.species == species]
    plt.scatter(subset['sepal length (cm)'],
                subset['sepal width (cm)'],
                c=color,
                label=species,
                alpha=0.7)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Sepal Length vs Width by Species')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.savefig('sepal_scatter.png', dpi=300, bbox_inches='tight')
plt.close()
```

Listing 12: Creating scatter plots

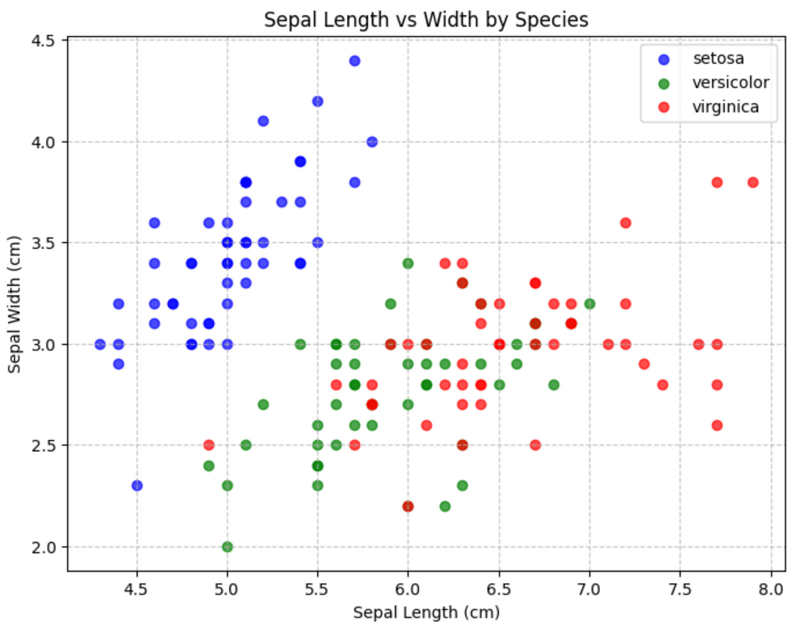

Figure 4: Scatter plot of sepal length vs width by species

7 Advanced pandas Techniques

7.1 Pivot Tables

```python
# Create a pivot table
pivot = pd.pivot_table(df,
                       values=['sepal length (cm)', 'petal length (cm)'],
                       index='species',
                       aggfunc=['mean', 'std'])
print(pivot)
```

Listing 13: Creating pivot tables

The pivot table would look like:

Table 5: Pivot table of sepal and petal length by species

	mean		std	
	sepal length (cm)	petal length (cm)	sepal length (cm)	petal length (cm)
species				
setosa	5.006	1.462	0.352	0.174
versicolor	5.936	4.260	0.516	0.470
virginica	6.588	5.552	0.636	0.552

7.2 MultiIndex and Advanced Indexing

```python
# Create a MultiIndex DataFrame
arrays = [
    ['setosa', 'setosa', 'versicolor', 'versicolor', 'virginica', 'virginica'],
    ['sepal', 'petal', 'sepal', 'petal', 'sepal', 'petal']
]
index = pd.MultiIndex.from_arrays(arrays, names=('species', 'part'))
columns = ['length', 'width']
data = [
    [5.006, 3.428], [1.462, 0.246],
    [5.936, 2.770], [4.260, 1.326],
    [6.588, 2.974], [5.552, 2.026]
]
multi_df = pd.DataFrame(data, index=index, columns=columns)
print(multi_df)
```

Listing 14: Using MultiIndex

This produces a MultiIndex DataFrame:

Table 6: MultiIndex representation of Iris measurements

species	part	length	width
setosa	sepal	5.006	3.428
	petal	1.462	0.246
versicolor	sepal	5.936	2.770
	petal	4.260	1.326
virginica	sepal	6.588	2.974
	petal	5.552	2.026

7.3 Applying Functions with apply()

```python
# Define a custom function to calculate the ratio
def calculate_ratio(row):
    return row['petal length (cm)'] / row['sepal length (cm)']

# Apply the function to each row
df['petal_sepal_ratio'] = df.apply(calculate_ratio, axis=1)

# Calculate summary statistics by species
ratio_by_species = df.groupby('species')['petal_sepal_ratio'].describe()
print(ratio_by_species)
```

Listing 15: Using apply() for custom functions

The results would look like:

Table 7: Summary statistics of petal-to-sepal length ratio by species

species	count	mean	std	min	25%	50%	75%	max
setosa	50	0.293	0.039	0.209	0.269	0.291	0.315	0.378
versicolor	50	0.717	0.050	0.581	0.686	0.720	0.750	0.815
virginica	50	0.842	0.049	0.738	0.810	0.845	0.877	0.958

8 Exporting Data for LaTeX

8.1 Converting DataFrames to LaTeX Tables

pandas provides built-in functionality to convert DataFrames to LaTeX tables:

```
# Convert DataFrame to LaTeX table
latex_table = df.head().to_latex(index=False)
print(latex_table)

# Save to file
with open('iris_table.tex', 'w') as f:
    f.write(latex_table)
```

Listing 16: Converting DataFrames to LaTeX

8.2 Customizing LaTeX Output

```
# Customize LaTeX output
latex_table = df.head().to_latex(
    index=False,
    float_format="%.2f",
    label="tab:iris_sample",
    caption="Sample of the Iris dataset",
    position="htbp",
    column_format="lrrrrl"
)
print(latex_table)
```

Listing 17: Customizing LaTeX output

9 Working with Missing Data

Although the Iris dataset doesn't contain missing values, handling missing data is a common task in data analysis:

```
# Create a copy with some missing values
df_missing = df.copy()
df_missing.loc[0:5, 'sepal length (cm)'] = np.nan
df_missing.loc[10:15, 'petal width (cm)'] = np.nan

# Check for missing values
print(df_missing.isnull().sum())
```

```

# Drop rows with missing values
df_no_missing = df_missing.dropna()

# Fill missing values with mean
df_filled = df_missing.fillna(df.mean())

# Fill missing values with different methods per column
df_filled2 = df_missing.fillna({
    'sepal length (cm)': df['sepal length (cm)'].mean(),
    'petal width (cm)': df['petal width (cm)'].median()
})
```

Listing 18: Handling missing data

10 Merging and Joining DataFrames

```
# Create two DataFrames
measurements = df[['sepal length (cm)', 'sepal width (cm)',
                   'petal length (cm)', 'petal width (cm)']]
measurements.index.name = 'id'
measurements = measurements.reset_index()

species_df = df[['species']].copy()
species_df.index.name = 'id'
species_df = species_df.reset_index()

# Merge DataFrames
merged_df = pd.merge(measurements, species_df, on='id')
print(merged_df.head())

# Different types of joins
left_join = pd.merge(measurements, species_df, on='id', how='left')
right_join = pd.merge(measurements, species_df, on='id', how='right')
outer_join = pd.merge(measurements, species_df, on='id', how='outer')
```

Listing 19: Merging DataFrames

11 Advanced Filtering and Selection

```
# Filter with complex conditions
large_sepals = df[(df['sepal length (cm)'] > 6.0) &
                  (df['sepal width (cm)'] > 3.0)]
print(large_sepals.head())

# Use query method
large_sepals_query = df.query('`sepal length (cm)` > 6.0 and `sepal width (cm)` >
    3.0')
print(large_sepals_query.head())

# Select with loc and iloc
# loc uses labels
subset_loc = df.loc[10:15, ['sepal length (cm)', 'species']]
```

```python
print(subset_loc)

# iloc uses integer positions
subset_iloc = df.iloc[10:15, [0, 4]]
print(subset_iloc)
```

Listing 20: Advanced filtering

12 Pandas with Iris Dataset: Case Studies

12.1 Case Study 1: Species Classification

```python
# Create a model based on simple thresholds
def classify_iris(row):
    if row['petal length (cm)'] < 2.5:
        return 'setosa'
    elif row['petal width (cm)'] < 1.8:
        return 'versicolor'
    else:
        return 'virginica'

# Apply the classification function
df['predicted_species'] = df.apply(classify_iris, axis=1)

# Calculate accuracy
accuracy = (df['species'] == df['predicted_species']).mean()
print(f"Classification accuracy: {accuracy:.2f}")

# Create a confusion matrix
conf_matrix = pd.crosstab(df['species'], df['predicted_species'],
                          rownames=['Actual'], colnames=['Predicted'])
print(conf_matrix)
```

Listing 21: Species classification with basic metrics

12.2 Case Study 2: Feature Engineering

```python
# Create new features
df['sepal_area'] = df['sepal length (cm)'] * df['sepal width (cm)']
df['petal_area'] = df['petal length (cm)'] * df['petal width (cm)']
df['sepal_perimeter'] = 2 * (df['sepal length (cm)'] + df['sepal width (cm)'])
df['petal_perimeter'] = 2 * (df['petal length (cm)'] + df['petal width (cm)'])
df['ratio_areas'] = df['petal_area'] / df['sepal_area']

# Check which features best separate the species
for feature in ['sepal_area', 'petal_area', 'sepal_perimeter',
                'petal_perimeter', 'ratio_areas']:
    print(f"\nFeature: {feature}")
    print(df.groupby('species')[feature].describe()[['mean', 'std']])
```

Listing 22: Feature engineering

13 Conclusion

This guide has covered a comprehensive set of pandas techniques for working with the Iris dataset, from basic data manipulation to advanced analytics. The examples provided showcase how to analyze, visualize, and export data in ways that are compatible with LaTeX documents. By integrating pandas with LaTeX, researchers and data scientists can create highly professional reports and publications that combine statistical analysis with proper documentation.

A Complete Code Example

Here's a complete example that combines many of the techniques discussed in this guide:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets

# Load the Iris dataset
iris = datasets.load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)

# Basic exploratory analysis
print("Dataset shape:", df.shape)
print("\nFirst 5 rows:")
print(df.head())
print("\nDataset info:")
print(df.info())
print("\nSummary statistics:")
print(df.describe())

# Feature engineering
df['sepal_area'] = df['sepal length (cm)'] * df['sepal width (cm)']
df['petal_area'] = df['petal length (cm)'] * df['petal width (cm)']
df['petal_sepal_ratio'] = df['petal length (cm)'] / df['sepal length (cm)']

# Group statistics
print("\nMean values by species:")
species_means = df.groupby('species').mean()
print(species_means)

# Visualization
plt.figure(figsize=(10, 8))
sns.pairplot(df, hue='species')
plt.savefig('iris_pairplot.png', dpi=300, bbox_inches='tight')
plt.close()

# Create a scatter plot with sepal vs petal area
plt.figure(figsize=(8, 6))
for species, color in zip(df['species'].unique(), ['blue', 'green', 'red']):
    subset = df[df['species'] == species]
    plt.scatter(subset['sepal_area'], subset['petal_area'],
                c=color, label=species, alpha=0.7)
plt.xlabel('Sepal Area (cm )')
```

```python
plt.ylabel('Petal Area (cm )')
plt.title('Sepal Area vs Petal Area by Species')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.savefig('area_scatter.png', dpi=300, bbox_inches='tight')
plt.close()

# Export results to LaTeX
with open('iris_analysis.tex', 'w') as f:
    f.write("\\section{Iris Dataset Analysis Results}\n\n")

    # Write summary statistics
    f.write("\\subsection{Summary Statistics}\n")
    f.write(df.describe().to_latex(float_format="%.3f"))

    # Write species means
    f.write("\n\\subsection{Mean Values by Species}\n")
    f.write(species_means.to_latex(float_format="%.3f"))

    # Include figures
    f.write("\n\\subsection{Visualizations}\n")
    f.write("\\begin{figure}[htbp]\n")
    f.write("    \\centering\n")
    f.write("    \\includegraphics[width=0.8\\textwidth]{iris_pairplot.png}\n")
    f.write("    \\caption{Pairplot of Iris features by species}\n")
    f.write("    \\label{fig:pairplot}\n")
    f.write("\\end{figure}\n\n")

    f.write("\\begin{figure}[htbp]\n")
    f.write("    \\centering\n")
    f.write("    \\includegraphics[width=0.7\\textwidth]{area_scatter.png}\n")
    f.write("    \\caption{Scatter plot of sepal area vs petal area by species}\n")
    f.write("    \\label{fig:area_scatter}\n")
    f.write("\\end{figure}\n")

print("Analysis complete. Results exported to LaTeX files.")
```

Listing 23: Complete analysis workflow