Chapters 2 and 3: Python Workbook

Ahmad Tahmid

Contents

1 Chapter 2 Solutions

1.1 Exercise 2.4.1

Goal

Perform basic Python operations: define arrays, slice and index them, create plots, and practice with fundamental Python commands.

Solution

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 1. Basic Numpy array creation
x = np.array([1, 2, 3, 4, 5])
y = x * 2

# Display them
print("x:", x)
print("y:", y)

# 2. Indexing
print("First element of x:", x[0])    # zero-based indexing
print("Last element of y:", y[-1])   # negative index gets last
    element

# 3. Simple Plot
plt.plot(x, y, 'o-r')  # red circles with a line
plt.title("Example Plot of x vs. y")
plt.xlabel("x")
plt.ylabel("y = 2*x")
plt.show()

# 4. Summaries with Numpy
print("Mean of x:", np.mean(x))
print("Std of y:", np.std(y))
```

1.2 Exercise 2.4.2

Goal

Load a dataset (e.g., `Auto.csv`) using Python, compute summary statistics, and make a scatter plot.

Solution

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

Auto = pd.read_csv("Auto.csv", na_values='?').dropna()

# Inspect the first few rows
print(Auto.head())

# Summary
print(Auto.describe())

# Example: scatter plot of mpg vs horsepower
plt.scatter(Auto["horsepower"], Auto["mpg"], alpha=0.5)
plt.xlabel("Horsepower")
plt.ylabel("MPG")
plt.title("Scatter Plot of MPG vs. Horsepower")
plt.show()

# Additional manipulation: power_to_weight = horsepower / weight
if "weight" in Auto.columns:
    Auto["power_to_weight"] = Auto["horsepower"] / Auto["weight"]
    print(Auto[["power_to_weight"]].head())
```

1.3 Exercise 2.4.3

Goal

Use slice notation, logical conditions, or loops to filter and process data.

Solution

```
import pandas as pd
import numpy as np

Auto = pd.read_csv("Auto.csv", na_values='?').dropna()

# 1. Subset rows with horsepower > 100
high_hp = Auto[Auto["horsepower"] > 100]
print("Number of cars with horsepower > 100:", len(high_hp))

# 2. Grouping by cylinders and finding average mpg
if "cylinders" in Auto.columns:
    grp = Auto.groupby("cylinders")
    mean_by_cyl = grp["mpg"].mean()
    print("Average MPG by cylinder count:\n", mean_by_cyl)
```

```
# 3. Simple loop example
mpg_list = Auto["mpg"].values
count_above_30 = 0
for val in mpg_list:
    if val > 30:
        count_above_30 += 1

print("Number of cars with mpg > 30:", count_above_30)

# 4. Label efficiency based on mpg
def label_efficiency(mpg):
    if mpg > 30:
        return "High"
    elif mpg > 20:
        return "Medium"
    else:
        return "Low"

Auto["efficiency_label"] = Auto["mpg"].apply(label_efficiency)
print(Auto[["mpg", "efficiency_label"]].head())
```

1.4 Exercise 2.4.4

Goal

Create histograms, boxplots, or other graphical summaries to compare distributions across groups.

Solution

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

Auto = pd.read_csv("Auto.csv", na_values='?').dropna()

# 1. Histogram of mpg
plt.hist(Auto["mpg"], bins=15, color='skyblue', edgecolor='black')
plt.xlabel("MPG")
plt.ylabel("Frequency")
plt.title("Histogram of MPG")
plt.show()

# 2. Boxplot of mpg by cylinders
if "cylinders" in Auto.columns:
    Auto.boxplot(column="mpg", by="cylinders")
    plt.title("MPG by Cylinders")
```

```
    plt.suptitle("")  # remove default title
    plt.xlabel("Cylinders")
    plt.ylabel("MPG")
    plt.show()

# 3. Pairplot with seaborn
sns.pairplot(Auto[["mpg","horsepower","weight","acceleration"]])
plt.show()
```

2 Chapter 3 Solutions

2.1 Exercise 3.7.1

Goal

Fit a simple linear regression model to predict `mpg` from `horsepower` in the `Auto` dataset. Examine coefficients, confidence intervals, and residual plots.

Solution

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

Auto = pd.read_csv("Auto.csv", na_values='?').dropna()

X = Auto[['horsepower']]
y = Auto['mpg']

X_const = sm.add_constant(X)
model = sm.OLS(y, X_const).fit()
print(model.summary())

# Confidence intervals
print("Confidence intervals:\n", model.conf_int())

# Plot regression
plt.scatter(Auto['horsepower'], Auto['mpg'], alpha=0.5)
hp_seq = np.linspace(X['horsepower'].min(), X['horsepower'].max(),
   100)
hp_seq_const = sm.add_constant(hp_seq)
mpg_pred = model.predict(hp_seq_const)
plt.plot(hp_seq, mpg_pred, color='red', linewidth=2)
plt.xlabel("Horsepower")
plt.ylabel("MPG")
```

```
plt.title("Linear Regression: MPG ~ Horsepower")
plt.show()

# Residual plot
residuals = model.resid
fitted = model.fittedvalues

plt.scatter(fitted, residuals, alpha=0.5)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Residuals vs. Fitted")
plt.show()
```

2.2 Exercise 3.7.2

Goal

Fit a multiple linear regression model (e.g., `mpg` predicted by `horsepower`, `weight`, and `year`), interpret the coefficients, and check model diagnostics.

Solution

```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

Auto = pd.read_csv("Auto.csv", na_values='?').dropna()

predictors = ["horsepower", "weight", "year"]
X_mult = Auto[predictors]
y_mult = Auto["mpg"]
X_mult_const = sm.add_constant(X_mult)

mlr_model = sm.OLS(y_mult, X_mult_const).fit()
print(mlr_model.summary())

# Confidence intervals
print("Confidence intervals:\n", mlr_model.conf_int())

# Residual vs Fitted Plot
residuals = mlr_model.resid
fitted = mlr_model.fittedvalues

plt.scatter(fitted, residuals, alpha=0.5)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel("Fitted Values")
```

```
plt.ylabel("Residuals")
plt.title("Multiple Regression Residuals vs. Fitted")
plt.show()

# Check correlation among predictors
corr_matrix = Auto[predictors].corr()
print("Correlation matrix:\n", corr_matrix)
```

2.3 Exercise 3.7.3

Goal

Fit a linear model with interaction terms or polynomial terms, and assess significance.

Solution

```
import pandas as pd
import statsmodels.formula.api as smf
import numpy as np
import matplotlib.pyplot as plt

Auto = pd.read_csv("Auto.csv", na_values='?').dropna()

# Interaction example: mpg ~ horsepower * weight
mlr_interact = smf.ols("mpg ~ horsepower * weight", data=Auto).fit()
print(mlr_interact.summary())

# Polynomial term: mpg ~ horsepower + horsepower^2
mlr_poly = smf.ols("mpg ~ horsepower + I(horsepower**2)", data=Auto)
   .fit()
print(mlr_poly.summary())

# Visualization of polynomial fit
hp_range = np.linspace(Auto["horsepower"].min(), Auto["horsepower"].
   max(), 100)
tmp_df = pd.DataFrame({"horsepower": hp_range})
preds_poly = mlr_poly.predict(tmp_df)

plt.scatter(Auto["horsepower"], Auto["mpg"], alpha=0.5)
plt.plot(hp_range, preds_poly, color='red', linewidth=2)
plt.xlabel("Horsepower")
plt.ylabel("MPG")
plt.title("Polynomial Fit: MPG ~ Horsepower^2")
plt.show()
```

2.4 Exercise 3.7.4

Goal

Incorporate a qualitative predictor in a linear regression model.

Solution

```
import pandas as pd
import statsmodels.formula.api as smf

Auto = pd.read_csv("Auto.csv", na_values='?').dropna()

# Convert 'origin' to categorical (example)
if Auto['origin'].dtype != 'object':
    Auto['origin'] = Auto['origin'].astype('category')

cat_model = smf.ols("mpg ~ horsepower + origin", data=Auto).fit()
print(cat_model.summary())

# Check how 'origin' was coded
print("Origin categories:", Auto['origin'].cat.categories)

# Example: setting 'usa' as baseline
Auto['origin'] = Auto['origin'].cat.reorder_categories(['usa','japan
    ','europe'])
cat_model2 = smf.ols("mpg ~ C(origin, Treatment('usa')) + horsepower
    ", data=Auto).fit()
print(cat_model2.summary())
```

3 My Learnings

- Worked with fundamental Python commands for data handling, plotting, and indexing
- read CSV files, handle missing data, and create summaries and visualizations (such as histograms, scatter plots, and boxplots).
- `statsmodels` to fit both simple and multiple regression models, interpret outputs, and visualize residuals.
- incorporate interaction effects, polynomial terms, and categorical predictors into regression models, and then interpret the results in terms of significance and model fit.
- experience with the Python data science stack (`numpy`, `pandas`, `matplotlib`, `seaborn`, and `statsmodels`) hmmmm