## Objectives

Estimated Learning Time: 3h 36m

**RETRIEVING DATA**

Good data is the fuel that powers Machine Learning and Artificial Intelligence. In this module you will learn how to retrieve data from different sources, how to clean it to ensure its quality, and how to conduct exploratory analysis to visually confirm it is ready for machine learning modeling.

Learning Objectives:

* Retrieve data from multiple data sources: SQL, NoSQL databases, APIs, Cloud
* Describe and use common feature selection and feature engineering techniques
* Handle categorical and ordinal features, missing values
* Use a variety of techniques for detecting and dealing with outliers
* Articulate why feature scaling is important and use a variety of scaling techniques

RETRIEVING DATA

Goals:

Retrievin gdata from mult data sources;

* Sql
* No sql
* Apis
* Cloud

Readings csv files

Csv – comma-separated files consist of rows or data, separated by commas,

In pandas, csv files can typically be read using just a few lines of code

Import pandas as pd

Filepath = ‘data/iris\_data.csv’

#import the data

Data = pd.read\_csv(filepath)

#Print a few rows

Print(data.iloc[:5]

# different delimiters – tab separated files (.tsv):

Data = pd.read\_csv(filepath, sep=’\t’)

# different dilimeters – space-separated files:

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])

Json – javascript object notation – files are a standard way to store data across platforms. Json files are very similar in structure to python dictionaries.

Reading json files into python:

# read json file as dataframe:

Data = pd.read\_json(filepath)

#write dataframe file to json

Data.to\_json(‘outputfile.json’)

Sql databases

Structured query language (swl) represent a set of relational databases with fixed schemas

There are many types of sql databases, which function similarly (with some subtle differences in syntax)

Examples of sql databases:

* Microsoft sql server
* Postgres
* Mysql
* Aws redshift
* Oracle db
* Db2 family

Reading sql data

#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\_swl(query, con)

While this example uses sqlite3, there are several other packages available

The sql module creates a connection with the database

Data is read into pandas by combining a query with this connection.

Nosql databases

Not-only sql (nosql) databases are not relational, vary more in structure. Depending on application may perform more quickly or reduce technical overhead.

Most nosql databases store data in json format

Examples of nosql databases:

* Document database: mongodb, couchdb
* Key-value stores: Riak, voldemort, redis
* Graph databases: neo4j, hypergraph
* Wide-column stores: cassandra, hbase

The below example uses the pymongo module to read files stored in mongodb, although there are several other packages available.

We first make a connection with the database (mongodb needs to be running)/

Data is read into pandas by combining a query with this connections

Here, query should be replaced with a mongoDB query string or {} to selectall)

# sql data imports

From pymongo import mongoclient

#create a mongo connection

Con = MongoClient()

# Choose database (con.list\_database\_names() will display available database)

Db = con.database\_name

# create a cursor object using a query

Cursor = db.connection\_name.find(query)

# expand cursor and construct dataframe

Df = pd.DataFrame(list(cursor))

A variety of data providers make data available via application programming interfaces (api’s), that makes it easy to access such data via python.

There are also a number of datasets available online in various formats.

An online available example is the uc irvine machine learning library.

Here, we read one of the datasets into pandas directly via the url.

#uci cars data set – url location

Data url = “http:// “

#read data into pandas

Df=pd.read\_csv(data\_url, header=None)

## Running Course Notebooks in Watson Studio on the IBM Cloud

## **Version Date: 2021 Mar 15**

All of the notebooks in these courses are written to run locally on your computer running a Jupyter notebook server. If you wish to run the notebooks in Watson Studio in the IBM Cloud, you will need to add some modifications to each notebook.

Why? Because once you import a course notebook and the data files for that notebook into a Watson Studio project, the data files are no longer available to the notebook!

This is simply due to the fact that the imported data files are stored in an IBM Cloud Object Storage (COS) bucket. The notebook does not have access to those objects in the COS bucket. Thus, if you import a notebook and its data files into a Studio project then try to run it, the notebook will return "File not found" errors.

In order to make the data files available to your notebook, you will need to run some code in your notebook to:

* Access the correct COS bucket
* Read your data file from the bucket into a byte stream object
* Write that byte stream object to the virtual disk of the container running the notebook.

**This whole process LOOKS complicated but it really isn't. After you've done it a couple of times, it can be done quickly and easily without looking at the instructions.**

#### **Before you begin**

What you'll be doing here is putting code into your notebook that simply gets a named file from a Cloud Object Storage (COS) bucket, puts in a byte stream, then creates a directory and file to write that byte stream to.

The most confusing part of all of this is the relative locations of the directories that you are creating in the virtual disk used by your notebook. Keep these points in mind:

1. When a notebook is executing it is running in a directory named /home/dsxuser/work

.

2. When you use the functions given below they create the directories one level up in /home/dsxuser

.

3. The code in your notebooks should refer to the files using either a relative path (e.g., ../data/classic\_rock.db) or an absolute path (e.g., /home/dsxuser/data/classic\_rock.db ).

4. Most of the relative paths used in the notebooks for these courses should work if you use the functions as described in the example below.

**Steps for running notebooks in Watson Studio**

**Step 0:** These steps assume you have already done ALL of the following:

* Logged into IBM Cloud and created an instance of Watson Studio
* Created a new project in Watson Studio Cloud
* Imported a course notebook and data assets (e.g., CSV files, Python scripts, SQLite database files) into a Watson Studio Cloud project

**Step 1:** From the "**My Projects**" page in Watson Studio Cloud, click on the link for the project containing your notebook and data files

**Step 2:** On the project overview page, click on the link "**Settings**" located near the top center of the page.

**Step 3:** On the Settings page, scroll down to section "**Access Tokens**"

**Step 4:** On the right hand side of the Access Tokens section, click the + sign link for "**New Token"**

**Step 5:** On the **New Token popup**, enter a name for your token (e.g. "*Tokey McTokenface*") and set "**Access role for project**" to "Editor"

**Step 6:** Click the button "**Create**"

**Step 7:** Scroll back to the top of the page and click on link "**Assets**"

**Step 8:** On the Assets page locate the row listing the notebook you wish to work with and click on the **pencil icon** on the right side of the row to edit the notebook.

**Step 9:** On the project menu, locate the **More** menu item (indicated by three dots stacked on top of each other) near the top right of the page and click on it.

**Step 10:** Click on the menu item "**Insert Project Token**"

You should now see code inserted to your notebook that resembles this:

**Step 11:** Insert a new cell under the project token cell shown above and paste the following code into it:

# START CODE BLOCK  
# cos2file - takes an object from Cloud Object Storage and writes it to file on container file system.  
# Uses the IBM project\_lib library.  
# See <https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/project-lib-python.html># Arguments:  
# p: project object defined in project token  
# data\_path: the directory to write the file  
# filename: name of the file in COS  
  
  
import os  
def cos2file(p,data\_path,filename):  
 data\_dir = p.project\_context.home + data\_path  
 if not os.path.exists(data\_dir):  
 os.makedirs(data\_dir)  
 open( data\_dir + '/' + filename, 'wb').write(p.get\_file(filename).read())  
  
  
# file2cos - takes file on container file system and writes it to an object in Cloud Object Storage.  
# Uses the IBM project\_lib library.  
# See <https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/project-lib-python.html># Arguments:  
# p: prooject object defined in project token  
# data\_path: the directory to read the file from  
# filename: name of the file on container file system  
  
  
import os  
def file2cos(p,data\_path,filename):  
 data\_dir = p.project\_context.home + data\_path  
 path\_to\_file = data\_dir + '/' + filename  
 if os.path.exists(path\_to\_file):  
 file\_object = open(path\_to\_file, 'rb')  
 p.save\_data(filename, file\_object, set\_project\_asset=True, overwrite=True)  
 else:  
 print("file2cos error: File not found")  
# END CODE BLOCK

This code contains two user defined Python functions.

* cos2file: Get an asset stored in Cloud Object Storage (COS) and write the data to a disk file in the notebook container.
* file2cos: Get a disk file in the notebook container and write to an asset stored in Cloud Object Storage (COS)

The arguments for both of these functions are:

* p: The project object defined by the project token, called "project" by default.
* data\_path: the directory path relative to /home/dsxuser
* that will contain the files.
* filename: the filename that the data should be written to or read from.

**Step 12:** Create a new empty cell under the one created in Step 11 and enter code to call the function cos2file() to retrieve a file from Cloud Object Storage.

**Example**

You are given a notebook called **01a\_DEMO\_Reading\_Data.ipynb**. That notebook makes use of the file:

* **classic\_rock.db**: Referenced in the notebook as ../data/classic\_rock.db.

First, create the project in Watson Studio, import the notebook, store the file **classic\_rock.db** as assets in the same project, and create a new project token.

Open the notebook in Edit mode and insert the project token as described above, then paste the code from Step 11 above into a new cell under the one containing the project token.

Create another new cell under the Step 11 code cell and enter the following code:

cos2file(project, '/data', 'classic\_rock.db')

This call to the cos2file() function will create the directory /home/dsxuser/data

and write the file classic\_rock.db to the directory /home/dsxuser/

data.

**Handy Tip: Creating directories in Watson Studio**

Some notebooks in this course are written to store files in directories that don't exist yet in Watson Studio. If you ever need to create an empty directory that your notebook can store things to, just use the cos2file() function to store any of your data assets into a directory named according to what is needed.

For example, calling the function with cos2file(project, '/images', 'streams.csv') will create the directory /home/dsxuser/images and write the CSV file to it. But you can ignore that CSV file and the notebook can then write and read to the new /home/dsxuser/images directory.

Quiz:

Q1. Which statement about the pandas read\_csv function is true?

* It can read both tab-delimitedand space-delimited data

Q2. Which of the following is an examples of a file type that uses juavascript objectnotation json formatting

* Jupyter/python (ipynb files)

Q3. The data below appears in data.txt, and pandas has been imported. Which python command will read it correctly into a pandas dataframe?

63.03 22.55 39.83 22.23 12.908 -0.23AB

43.25 35.21 12.38 95.12 11.205 -0.35AB

- pandas.read\_csv(‘data.txt’, header=None, sep=’’)

**DATA CLEANING**

LEARNING Goals:

* why data cleaning is important for ml
* Issues that arise with messy data
* How to id duplicated or unnecessary data
* Policies for deal with outliers

Why is data cleaning so important

* Decisions and analytics are increasingly drive by data and models.
* Key aspects of ml workflow depend on cleaned data:
  + Obsercations: 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
* Messy data can lead to garbase in, garbage out

The main data problems companies face:

* Lack of data
* Too much data
* Bad data

Having data ready for ml and ai ensures you are read to infuse ai across your organization

How can data be messy:

* Duplicate or unnecessary data
* Inconsistent text and typos
* Missing data
* Outliers
* Data sourcing issues:
  + Mult. Systems
  + Different database types
  + On premises, in cloud
  + Etc.

Duplicated or unnecessary data

Pay attention to duplicate values and research why there are multiple values.

It’s a good idea to look at the features you’re bringing in and filter the data as necessary (be careful not to filter too much if you may use featuers later).

Handling missing values and outliers

Policies for missing data

* Remove the data; remove the row(s) entirely.
* Impute the data: replace with substituted values. Fill in the missing data with the most common value, the average value, etc.
  + Pro – not missing as much data
  + Con – adds level of uncertainty
* Mask the data: create a category for missing values.
  + Pro – don't lose rows, data
  + Add level of uncertainty

Outliers – is an observcation in data that is distant from most other observations.

Typically, these observations are aberrations and do not accurately represent the phenomenon we are trying to explain through the model.

If we do not id and deal with outliers they can have a significant impact on the model

Its important to remember that some outliers are informative and proovide insights into the data.

How to find outliers

Plots: historgram, density plot, box polot

Statistics: interguartile range, standard deviation

Residuals: standaradized, dleleted, studentized.

Detecting outliers: plots

#plot a histogram and density plot

Sns..displot(data, bins=20);

#boxplot

Sns.boxplot(data);

Detecting outliers: statistics

Import numpy as np

#calculate the interquartile range

Q25, q50, q75 = np.precentile(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)

#id the points

[x for x in data[‘Unemployment’] if x > max]

Detecting outliers: residuals

Residuals (difference between actual and predicted values of the outcome variable) represent model failure

Approaches to calculating residuals:

* Standardized – residual divided by standard error.
* Deleted: residual from fitting model on all data excluding current observation
* Studentized – deleted residuals divided by residual standard error (based on all data, or all data excvluding current observation.)

Policies for outliers

* Remove them.
  + Pro – they don’t effect
  + Lost data
* Assign the mean or median value
  + Don't have effect
  + Lost data
* Transform the variable
* Predict the what the value would be
  + Usinging similar observations to predict likely values.
  + Using regression.
* Keep them, but focus on model that are resistant to outliers.

Quiz  
outliers must be very extreme to noticeably impact the fit of a statistical model?

* False

Which residual-based approach to id outliers compares running a model with all data to running the same mode, but dropping a single observation

* Externally-studentized residuals

Outliers shjould always be replaced, since they never contain useful information about the data

* False

**EXPLORATORY DATA ANALYSIS**

EDA learning goals

Eda – is an approach to analayzing data sets to summarize their main characteristics, often with visual methods

Why is eda useful

Eda allows us to get an intial feel for the data

This lets us determine if the data meakes sense, or if further cleaning or more data is needed.

Eda helps to id patterns and trends in the data.

Techniques for eda:

* Summary statitics:
  + Average, median, min, max, correlations, etc.
  + Visualizations:
  + Histograms, scatter plots, box plots, etc

Tools for eda

Data wrangling:

* Pandas
* Visualization:
* Matplotlib, seaborn

Eda job applicant summary statistics:

Suppose you want to examine characterestics of job applicants:

Average: we could look at the average of all interview scores (perhaps by city or job function).

Max: we could look at most common words applicants used in application materials

Correlations: we could look at the correlations between technical assessments and years experience (perhaps by type of experience)

Sampling from dataframes:

# sample 5 rows without replacement

Sample = data.sample(n=5, replace=False)

Print(sample.iloc[:,-3:])

Replace = false is the default, so that data isn’t duplicated

-3: = last three columns

Theres many reason to consider random samples from dataframes:

* For large data, a random sample can make distribution easier.
* We may want to train models on a random sample of the data
* We may want to over/under sample observations when outcomes are uneven.

Visualization libraries

Visualizations can be created in mult ways:

* Matplotlib
* Pandas (via matplotlib)
* Seaborn
  + Statistically-focused plotting methods
  + Global preferences incorporated by matplotlib

Basic scatter plots with matplotlib

#panadas dataframe a[pproach

Import matplotlib.pyplot as plt

Plt.plot(data.sepal\_length, data.sepal\_width, ls =’’, marker=’o’)

Ls = line style

Marker is a marker

Scatter plots with multiple layers

# first plot statement

Plt.plot(data.sepal\_length, data.sepal\_width, ls =’’, marker=’o’, label=’sepal’)

#second plot statemetn

#panadas dataframe a[pproach

Plt.plot(data.pedal\_length, data.pedal\_width, ls =’’, marker=’o’, label=”petal”)

Historgrams

# pandas dataframe approach

Plt.hist(data.sepal\_length, bins=25)

Customizing plots

# matplotlib syntax

Fig, ax = plt.subplots()

Ax.barh(np.arrange(10), data.sepal\_width.iloc[:10])

# set position of ticks and tick labels

Ax.set\_yticks(np.arrange(0.4,10.4,1.0))

Ax.set\_yticklabels(np.arrange(1,11))

Ax.set(xlabel=”xlabel’, ylabel=’ylabel’, title=’Title’)

Customizing plots by group

# pandas dataframe approach

Data.groupby(‘species’).mean().plot(color=['red’, ‘blue’, ‘black’, ‘green’],fontsize=10.0, figsize=4,4)

Pair plots for features

#seaborn plot, feature correlations

Sns.pairplot(data, hue=’species’, size=3)

Seaborn example: hexbin plot

#seaborn hexbin pllot

Sns.joinplot(x=data[‘sepal\_length’], y=data[‘sepal\_width], king=’hex’)

Seaborn example: facet grid

#seaborn plot, facet grid

#first plot statement

Plot – sns.FacetGrid(data, col=’species’, margin\_titles=True)

Plot.map(plt.hist, ‘sepal\_eidth’, color=’green’)

#second plot statement

Plot = sns.FacetGrid(data, col=’species’, margin\_titles=True)

Plot.map(plt.hist, ‘sepal\_length’, color=’blue’)

**Feature Engineering and Variable Transformation**

Goals:

* Feature engineering and variable transformation
* Feature encoding
* Feature scaling

Transforming data

Models used in ml workflows often make assumptions about the data

A common ex. Is the linear regression model. This asssumes a linear relationship observations between and target (outcomes) variables.

Transformation of data distributions

Predictions from linear regression models assume residuals are normally distributed.

Features and predicted data are often skewed (distorted away from the center).

Data transformations can solve this issue.

# useful transformation functions

From numpy import log, loglp

From scipy.stats import boxcox

Log transformation ex.

Ex of right (positive) skewed

#plot a histogram and density plot

Sns.distplot(data, bins=20)

Ex after data transformation to have a normal skew

Import math

Log\_data = [math.log(d) for d in data[‘Unemployment’])

#plot transforme dplots

Sns.distplot(log\_data, bins=20)

Transformations: log features

Log transformations can be useful for linear regression.

The linear regression model involves linear combinations of features.

Transformations: polynomial features

We can estimate higher-order relationships in this data by adding polynomial features.

This allows us to use the same linear model.

Even with higher-order polynomials

Polynomial features: syntqax

# import the class containing the transformation method

From sklearn.preprocessing import polynomialfeatures

#create an instance of the class (choose number of degrees)

Polyfeat = polynomialfeatures(degree=2)

#create the polynomial features and then transform the data

Polyfeat = polyfeat.fit(X\_data)

X\_poly = polyFeat.transform(DX\_data)

Pt2.

Variable selection:

Variabable selection involves choosing the set of features to include in the model.

Variables must often be transformed before they can be included in models. In addition to log and polynomial transformatioons often involve:

* Encoding – convertin non-numerical features into numeric features
* Scaling – convertin the scale of numeric data so they are comparab le.

The appropriate method of scaling or encoding depends on the type of feature.

Feature encoding; types of features

Encoding is often applied to categorical features that take non-numeri values.

2 primary types

* Nominal – categorical variables that take in unordereed categories (e.g. red, blue, green; true, false)
* Ordinal – categorical variables that take values in ordered categoeries (high, low, med.)

Feature encoding approaches

There are several common approaches to eencoding variables

* Binary encoding converts variables to either 0 or 1 and suitable for variables that tak two possible values (e.g. true, false)
* One-hot encoding converts variables that take mult. Values into binary (0,1) variables, one for each category. This creates several new variables.
* Ordinal encoding involves convertin ordered categories to numerical values, usually by creating one variable that takes integer equal to the number of categories (e.g. 0, 1, 2, 3, …)

Feature scaling

Feature scaling involves adjusting a variables scale this allows comparison of variables with diff. Scales.

Diff continuous (numeric) features often haee different scales.

Why this might be an issue?

Feature scaling approaches

There are many approaches to scaling features

Some of the more common approaches include:

* Standar scaling – converts features to standard normal variables (by subtracting the mean and dividing by the standard error)
* Min-max scaling – converts variables to continuous variables in the (o, 1 ) interval by mapping minimum values to 0 and maximum values to 1. This type of scaling is sensitive to outliers.
* Robust scaling – is similar to min-max scaling, but instead maps the interquartile range (the 75th percentile value minus the 25th percentile value) to (0,1). This means the variable itself takes values outside of the (0, 1) interval. \

Commmon variable transformations

|  |  |
| --- | --- |
| **Feature type** | **Transformation** |
| Continuous – numerical values | Standard, min-max, robust scaling |
| Nominal categorical, unordered features (t/f) | Binary, one-hot encoding(0,1) |
| Ordinal – categorical, ordered features (movie ratings) | Ordinal encoding (0, 1, 2, 3) |

**Solution: Feature Engineering Lab – Part 1**

Walk through of lab

Df.info yells us how many non-null values each of our columns has.

Using the historgram we can check for outliars

So we filter outvalues below 4000

Df = fdf.loc[df[‘gr liv area’] <= 4000,:]

Df.copy wie can save a copy of our original data

Len(Df.PID.unique()) checks unique values from PID

Because all of the values above are unique we will drop this and another column . To make sure we are looking at column index instead of row index we use the axis=1

Df.drop([‘PID’, ‘Order’],axis=1,inplace=true)

To pull out numerical dtypes we use df.select\_dtypes(‘number’).columns

Above 0 has a right skew

Below 0 has a left skew

Above .75 is heavily skewed distribution

0 means no skew

Plt.subplots graph 2 axis and 1 figures

Df[field].hist(ax=ax\_before) -> to use pandas ploting

Set column equal to the log of itself with following:

Df[col] = df[col]

Lab solution pt. 2

Data.isnull().sum() -> true values will be one and false values will be 0

To transpose a dataset use .T

.info

.describe

Will all null values with 0

Smaller\_df smaller\_df.fillna(0)

Pair plot function from seaborn

These plots tell us about our target.

We want to insure theres not to much of a multi-collinearity between each one of our features

Adding polynomial features

We want to copy x so that we have x available to us later if needed

Inputing categorical features as numerical values

We introduce dummy variables or one-hot encoding

Value\_counts() from pandas

Pd.get\_dummies

Every columns that’s an object will come up with a variable

.transform() -> allows us to come up a value for every single row

Polynomial features in scikit-learn – will be important in everyday work with machine learning

* Quiz

Classification models require that input features be scaled.

* False

Feature scaling allows better interpretation of distance-based approaches

* True

Feature scaling reduces distortions caused by cariables with different scales.

* True

## End of module review: Exploratory Data Analysis for Machine Learning

### **Retrieving Data**

You can retrieve data from multiple sources:

* SQL databases
* NoSQL databases
* APIs
* Cloud data sources

The two most common formats for delimited data flat files are comma separated (csv) and tab separated (tsv). It is also possible to use special characters as separators.

SQL represents a set of relational databases with fixed schemas.

### **Reading in Database Files**

The steps to read in a database file using the sqlite library are:

* create a path variable that references the path to your database
* create a connection variable that references the connection to your database
* create a query variable that contains the SQL query that reads in the data table from your database
* create an observations variable to assign the read\_sql functions from pandas package
* create a tables variable to read in the data from the table sqlite\_master

JSON files are a standard way to store data across platforms. Their structure is similar to Python dictionaries.

NoSQL databases are not relational and vary more in structure. Most NoSQL databases store data in JSON format.

### **Data Cleaning**

Data Cleaning is important because messy data will lead to unreliable outcomes. Some common issues that make data messy are: duplicate or unnecessary data, inconsistent data and typos, missing data, outliers, and data source issues.

You can identify duplicate or unnecessary dataCommon policies to deal with missing data are: remove a row with missing columns, impute the missing data, and mask the data by creating a category for missing values.

Common methods to find outliers are: through plots, statistics, or residuals.

Common policies to deal with outliers are: remove outliers, impute them, use a variable transformation, or use a model that is resistant to outliers.

### **Exploratory Data Analysis**

EDA is an approach to analyzing data sets that summarizes their main characteristics, often using visual methods. It helps you determine if the data is usable as-is, or if it needs further data cleaning.

EDA is also important in the process of identifying patterns, observing trends, and formulating hypothesis.

Common summary statistics for EDA include finding summary statistics and producing visualizations.

### **Feature Engineering and Variable Transformation**

Transforming variables helps to meet the assumptions of statistical models. A concrete example is a linear regression, in which you may transform a predictor variable such that it has a linear relation with a target variable.

Common variable transformations are: calculating log transformations and polynomial features, encoding a categorical variable, and scaling a variable.

### Question 1

Correct

1.00 points out of 1.00

Flag question

#### Question text

Which of the following statements about cloud data access using Pandas is TRUE?

Select one:

A.

With read\_csv, the online file must be comma-delimited.

B.

The read\_csv function can read data directly from a website or url.

C.

With read\_csv , the destination file must have column names in the first row.

D.

A remote destination file must be downloaded locally before it can be read by Pandas.

#### Feedback

Correct

The correct answer is: The read\_csv function can read data directly from a website or url.

### Question 2

Correct

1.00 points out of 1.00

Flag question

#### Question text

In which case below is it most plausible to conclude that an observation includes an outlier for one of the features?

Select one:

A.

One feature has a deleted residual value above 3.

B.

The observation includes the maximum target value.

C.

The observation is missing values for several of the features.

D.

One feature has an internally-studentized residual value above 3.

#### Feedback

Correct!

The correct answer is: One feature has an internally-studentized residual value above 3.

### Question 3

Correct

1.00 points out of 1.00

Flag question

#### Question text

Which of these approaches to feature engineering will be impacted LEAST by extreme values?

Select one:

A.

RobustScaler

B.

MinMaxScaler

C.

LabelBinarizer

D.

OneHotEncoder

#### Feedback

Correct!

The correct answer is: RobustScaler

### Question 4

Correct

1.00 points out of 1.00

Flag question

#### Question text

Which of these approaches to feature engineering will be impacted MOST by extreme values?

Select one:

A.

RobustScaler

B.

MinMaxScaler

C.

LabelBinarizer

D.

OneHotEncoder

#### Feedback

Correct!

The correct answer is: MinMaxScaler

### Question 5

Correct

1.00 points out of 1.00

Flag question

#### Question text

(True/False) RobustScaler adapts MinMaxScaler to account for outliers.

Select one:

True

False

#### Feedback

Correct!

The correct answer is 'True'.

### Question 6

Correct

1.00 points out of 1.00

Flag question

#### Question text

(True/False) StandardScaler requires data that are normally distributed.

Select one:

True

False

#### Feedback

Correct!

The correct answer is 'False'.

### Question 7

Correct

1.00 points out of 1.00

Flag question

#### Question text

(True/False) Any features that have been transformed by StandardScaler, MinMaxScaler, or RobustScaler will take values in (0,1).

Select one:

True

False

#### Feedback

Correct!

The correct answer is 'False'.

### Question 8

Correct

1.00 points out of 1.00

Flag question

#### Question text

Which of the following assertions describes a good reason for using scatter plots to complement calculating the correlation coefficient between two variables?

Select one:

A.

A scatter plot helps you visualize whether outliers are inflating or deflating a correlation coefficient.

Correct!

B.

A scatter plot will help you identify if the correlation is positive or negative.

C.

A scatter plot takes into account both the spearman and the pearson correlation coefficients in a single step.

D.

It is computationally more efficient to produce a scatter plot first and then compute a correlation coefficient.

#### Feedback

Correct!

The correct answer is: A scatter plot helps you visualize whether outliers are inflating or deflating a correlation coefficient.