Assignment-1: Introduction to Virtualization and Service Orchestration

Instructions:

1. *This is an individual homework. Each student must submit his/her solution to the problem through Assignment-1 dropbox.*
2. All submissions are to be via the Quercus dropbox; no other way of submission is accepted.
3. All submissions must be submitted through the dropbox by the due date or by the closure date, Clouse Date allows 3 days for late submissions.

Purpose and Objective

This is an introductory assignment that addresses the basic concepts of cloud transformation.

Success Criteria

Successful identification of the concepts accurately as well as accurately demonstrating the ML Lab component.

Keywords

Virtualization, hypervisor, cluster, node, cell, fallback, container, data training, model, Image, container, volume, network, machine learning, SciPy, service, composition.

Readings

1. Architecture of Virtual Machines by R. P. Goldberg, https://www.cse.psu.edu/~buu1/teaching/spring06/papers/goldberg.pdf.
2. Formal Requirements for Virtualizalble Third Generation Architecture, Geral Popek,Robert Goldberg.
3. A virtual machine time-sharing system by R. A. Meyer and L. H. Seawright https://www.seltzer.com/margo/teaching/CS508.19/papers/meyer-1970.pdf
4. An efficient virtual machine implementation by RONALD J. SRODAWA and LEE A. BATES V/ayne State University, https://dl.acm.org/doi/pdf/10.1145/1499586.1499668
5. A PROGRAM SIMULATOR BY PARTIAL INTERPRETATION, Kazuhiro Fuchi, Hozumi Tanaka , Yuriko Manago and Toshitsugu Yuba , Electrotechnical Laboratory, Japanese Government, https://dl.acm.org/doi/pdf/10.1145/961053.961092
6. A Performance Comparison Between Enlightenment and Emulation in Microsoft Hyper-V By Hasan Fayyad-Kazan, Luc Perneel & Martin Timmerman: https://globaljournals.org/GJCST_Volume13/4-A-Performance-Comparison.pdf
7. Hyper V: https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-on-windows-server
    a. https://learn.microsoft.com/en-us/virtualization/hyper-v-on-

windows/reference/hyper-v-requirements.
   b. https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/
   c. https://learn.microsoft.com/en-us/system-center/vmm/?view=sc-vmm-2022
   d. https://learn.microsoft.com/en-us/windows-server/failover-clustering/failover-clustering-overview
   e. https://www.vmware.com/ca/products/aria.html
   f. https://www.vmware.com/ca/products/hyper-converged-infrastructure.html

8. Machine Learning Studio (Classic):
   a. Azure Machine Learning
      i. https://learn.microsoft.com/en-us/azure/machine-learning/migrate-overview
      ii. https://azure.microsoft.com/en-us/products/machine-learning/
   b. Documentation: https://learn.microsoft.com/en-us/previous-versions/azure/machine-learning/classic/

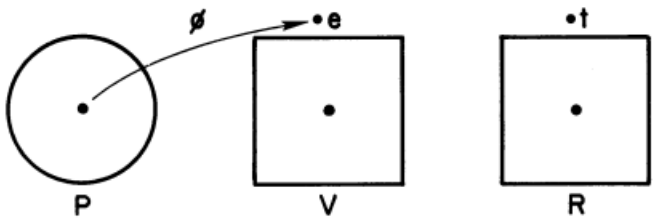
9. Docker Desktop and Docker Hub:
   a. https://www.docker.com/products/docker-desktop/
   b. https://hub.docker.com/
10. Docker Jupyter: https://jupyter-docker-stacks.readthedocs.io/en/latest/using/running.html
11. Docker documentation: https://docs.docker.com/engine/reference/commandline/cli/
12. Docker Language Specific Documentation: https://docs.docker.com/language/

13. SciPy Notebook: https://hub.docker.com/r/jupyter/scipy-notebook

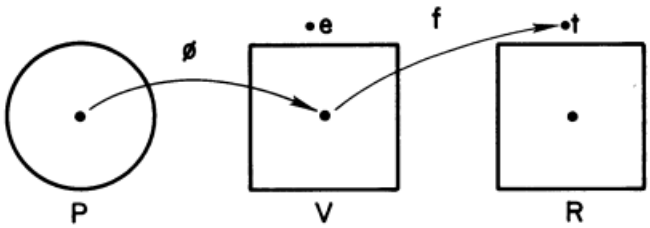
Q.1 Introduction to Virtualization [15 marks]

a. Based on the Goldberg Architecture of Virtual machine Model, answer the following questions:

1. How are resources addressed and abstracted?
2. Contrast between the Φ-map and the f-map.
3. Explain how virtual machine deployment facilitates new advantages that system programmers would achieve.
4. Identify one operating system that supported a virtual machine as well as three computer systems.
5. Interpret/explain the process map/state machine diagram displayed below:



(a)

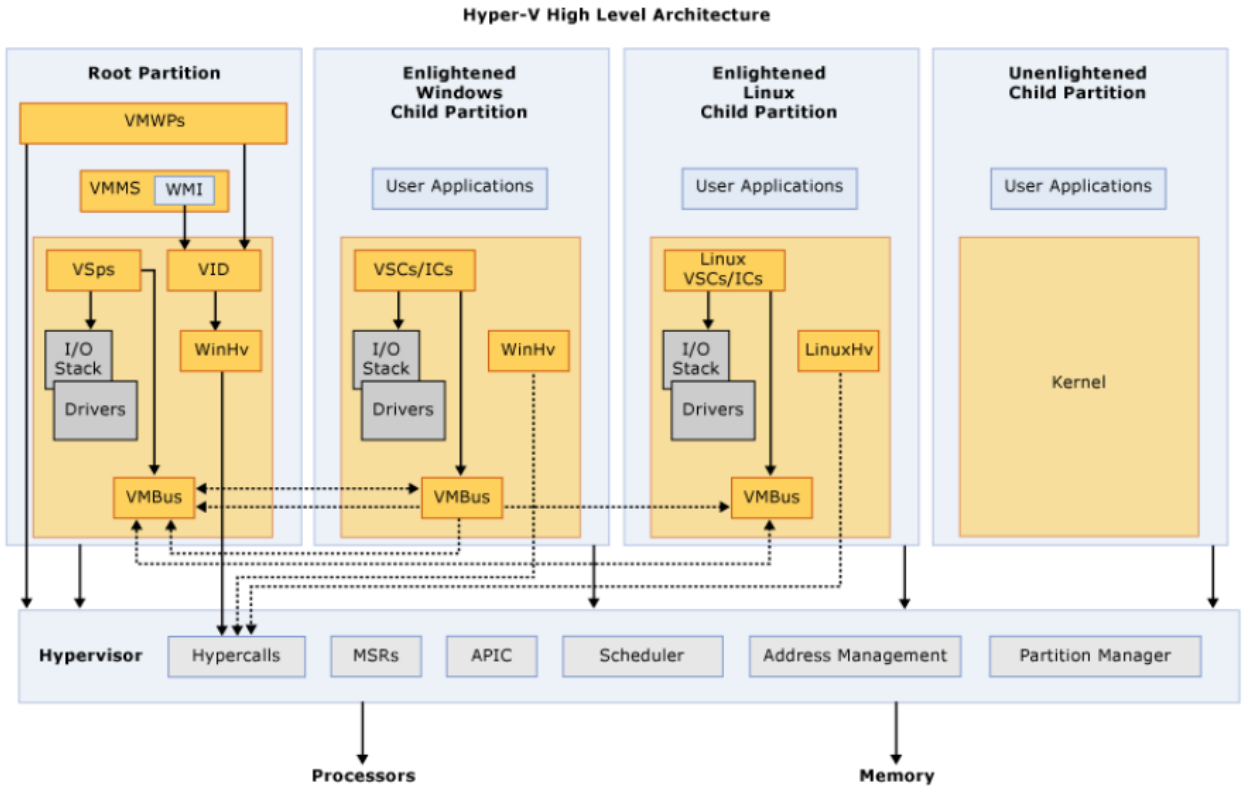


(b)

b. Based on Goldberg's paper "Formal Requirements for Virtualizable Third Generation Architectures" [2], what is a virtual machine is?
c. Based on [2], Formally identify the components of a virtual machine.
d. Based on [2], what are the elements of Program Status Word?
e. Based on [2], what is a Virtual Machine Monitor?
f. Based on [2], list the properties of virtual machines.
g. Based on [2], what is Recursive Virtualization?

Q.2 Virtualization Models [15 marks]

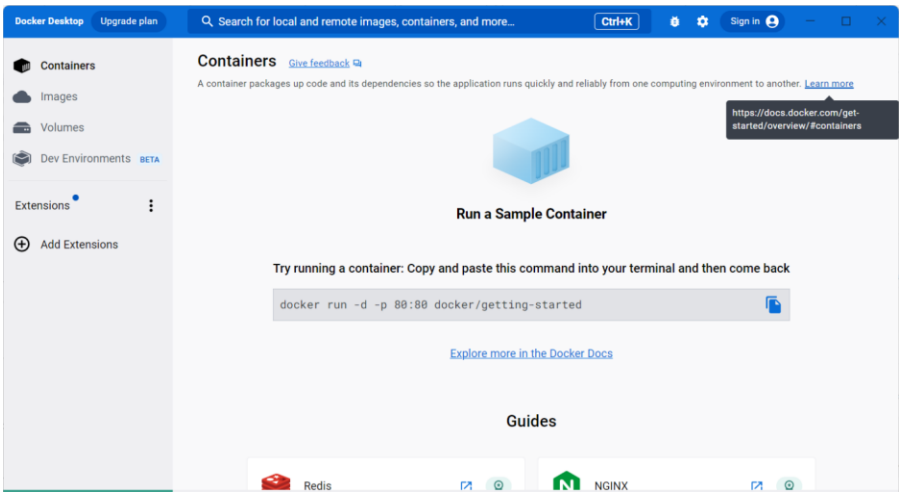
The Hyper-V lets you create a virtualized computing environment where you can create and manage virtual machines.



1.1 Does Hyper-V support type 1 virtualization or type 2 virtualization?
1.2 The paper [Timmerman et al] identifies performance metrics used for evaluation tests purposes, list those metrics.
1.3 What approaches/modes does Hyper-V support.
1.4 Explain how would paravirtualization differ from hardware emulation approaches.
1.5 How does Enlightened partition differ from Unenlighted partition?
1.6 What is the core requirement for Enlighted virtualization?
1.7 What is the core requirement for Emulated Virtualization?
1.8 what is failover clustering and identify why would recommend its deployment?
1.9 An associated software to Hyper-V is Virtual Machine Manager; explain what functionalities the software provides. Identify its capacity limits in terms of number of physical hosts, number of virtual machines, number of services, number of clouds, roles classifications, and number of logical networks.
1.10 VMware is one of the leading virtualization product providers; VMware HCI (Hyperconverged infrastructure) is a software-defined, unified system that combines all the elements of a traditional data center: storage, compute, networking and management. Identify why would your recommend such a scheme to your organization, and identify four Vmware products that would facilitates the deployment of and HCI system

Q.3 Basic Container Interaction [10 marks]

a. Create a Docker Hub account: https://hub.docker.com/signup Verify the account using the email sent to your email account.
b. Download and install Docker Desktop: https://www.docker.com/products/docker-desktop/
c. Start Docker Desktop



d. Open a command prompt session/terminal.
e. Run and document the output of the following commands, provide a screen snapshot for each command output:
   1. **docker version**
   2. **docker info**
   3. **docker login (use the credentials that you have specified in 1.a)**
   4. **docker images -a**
   5. **docker pull**
   6. Pull the Alpine Linux image from the hub: **docker pull alpine**
   7. Verify the image download: **docker images -a**
   8. Rename the image name from Alpine to Alpin500: **docker tag alpine:latest alpine500:latest**
   9. Verify the name update: **docker images -a**
   10. Run 3 Alpine container instances based on the Alpine500 image, where Alpin1 & Alpine2 are detached, and Alpine3 is in interactive mode
       a. **docker run --name** Alpine1 **-d** alpine500
       b. **docker run --name** Alpine2**-d** alpine500
       c. **docker run --name** Alpine3 **-it** alpine500
          i. list the contents of the /etc directory
             1. **cd /etc**
             2. **ls -l**

ii. List the available shells: cat /etc/shells

11. Keep the current window open and start a new command line session/terminal and list **all** containers: **docker container ls -a**
12. List all containers that are up and running: **docker container ls**
13. List the containers: **docker ps -a**
14. **Verify that all containers up an running:**

```
C:\Users\khali>docker ps -a
CONTAINER ID   IMAGE       COMMAND     CREATED         STATUS                      PORTS     NAMES
950c458ad4ec   alpine500   "/bin/sh"   9 minutes ago   Up 56 seconds                         Alpine3
2cb3d27110c2   alpine500   "/bin/sh"   9 minutes ago   Exited (0) 28 seconds ago             Alpine2
0415b8f0ca1e   alpine500   "/bin/sh"   9 minutes ago   Exited (0) 9 minutes ago              Alpine1
```

Provide a screen snapshot that shows Up for the three containers.

15. Pull the Redis in-memory key-value datastore engine (provide a screen snapshot):
    docker pull redis
16. List the images:
    docker images -a
17. Create two redis datastores:
    a. **docker run --name** redis0 **-d** redis
18. List all images: **docker images -a**
19. List all containers: **docker ps -a**
    a. **When was the redis0 container created?**
    b. **How can you access the shell terminal of the Linux Alpine OS?**
20. List the containers that are up and running: **docker ps**
21. **Instantiate a new standalone container of redis in interactive mode.**

Q.4 Image Composition, Service Orchestration, & Container Instantiation [25 Marks]

Using Docker Compose, compose the Flask-based image that conforms to the following specification:

The Image/Service Composition Process:

- Define the Application Dependency.
- Create a Dockerfile.
- Define Services in a Compose File.
- Build and Run Your App with Compose.

1. *Define the app.py script*

app.py

```python
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

2. *Define the application dependency in Requirements.txt*

requirements.txt

```
flask
redis
```

3. ***Create the Dockerfile*** *that conforms to the following:*

3.1. Build an image starting with the Python 3.7 image.
3.2. Set the working directory to /code.
3.3. Set environment variables used by the flask command.
3.4. Install gcc and other dependencies
3.5. Copy requirements.txt and install the Python dependencies.
3.6. Add metadata to the image to describe that the container is listening on port 10000

3.7. Copy the current directory . in the project to the workdir . in the image.
3.8. Set the default command for the container to flask run.

Dockerfile

```
1  # syntax=docker/dockerfile:1
2  FROM python:3.7-alpine
3  WORKDIR /code
4  ENV FLASK_APP=app.py
5  ENV FLASK_RUN_HOST=0.0.0.0
6  RUN apk add --no-cache gcc musl-dev linux-headers
7  COPY requirements.txt requirements.txt
8  RUN pip install -r requirements.txt
9  EXPOSE 5000
10 COPY . .
11 CMD ["flask", "run"]
```

4. ***Services definition in a Compose File: compose-docker.yml***

docker-compose.yml

```
1  version: "3.9"
2  services:
3    web:
4      build: .
5      ports:
6        - "8000:5000"
7    redis:
8      image: "redis:alpine"
```

The port binding should be 9001:5000, where the container port is 9001 and the host binding is 5000.

5. ***Build and run the service using the command: docker compose up***
6. ***Access the service using your browser: localhost:9001***
7. *Refresh the page several times and notice the counter being incremented. Document the value of the counter.*
8. *Open another terminal and list all running containers: docker container ls*
9. *Stop the containers by clicking stop in the server terminal for both containers.*
10. *Attempt to refresh the browser.*

11. *Start both of the containers using the command: docker container start container_name*
12. *Access the service using port 9001 and document the value of the counter.*
13. *Step 5 Push the image to the Docker Hub*
    - *13.1. Create a web version of the image and tag it:*
      docker tag image_id your_hub_name/hub_repository_name:tag

      Example: docker tag a5e0a24a719 abuosbak/ece1724h:ver1.0
      List the docker images and containers.

    - ***13.2.*** *Push image to the Docker Hub*
      docker push your_hub_name/hub_repository_name:tag

      Example: docker push abuosbak/ece1724h:ver1.0

14. *pull the following image to your Docker Desktop:*

    docker pull abuosbak/ece1724h:ver1.0
    attempt running the service and test it in the browser.

Q.5 Cloud Computing: Service Composition [15 marks]

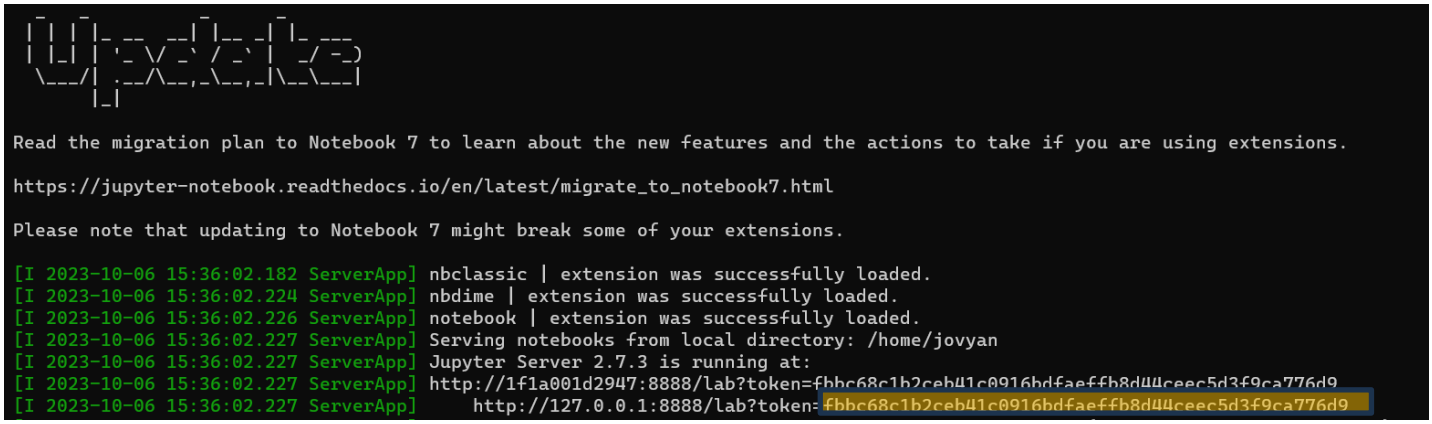
- Use the Docker Desktop to pull the SciPY: **docker pull jupyter/scipy-notebook**

- Create two instances of the container by running:
docker run -it -p 8888:8888 jupyter/scipy-notebook
and
docker run -it -p 8889:8888 jupyter/scipy-notebook

- Access the Jupyter service in your browser using the URL:

http://127.0.0.1:8889/lab?token=Token_Value
where token_value would be provided by the server in the execution flow.

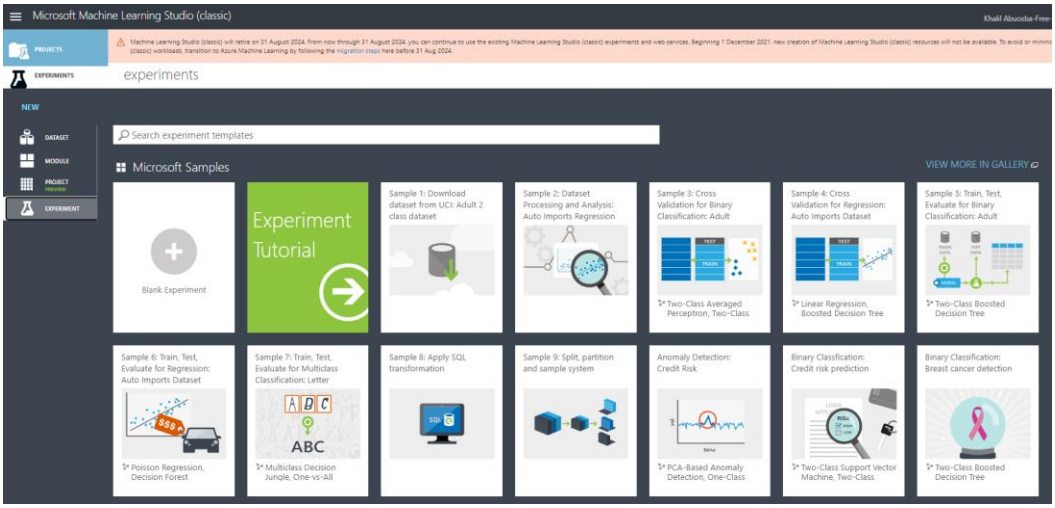
1. Start a Jupyter notebook.
2. Upload the dataset to the notebook (the abc.csv file).
3. Paste the contents of linReg.py file in a single cell or run each statement in a single cell.
4. What are the values of the coefficient and the y-intercept.
5. take a screen snapshot of the scatter diagram.



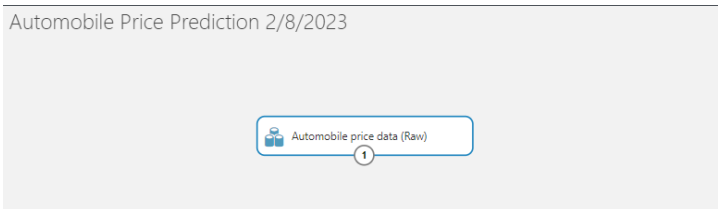
The Token would be generated at runtime.

Q.6 "Machine Learning as a service" Lab [20 marks]

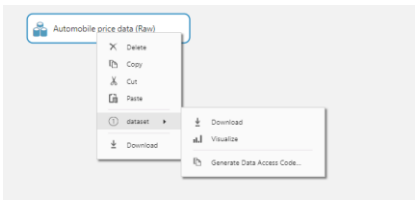
1. sign-in into Studio: https://studio.azureml.net/
2. In this lab, you will create a Machine Learning Studio experminment.
3. Click on Experiment
    a. Create a new experiment by clicking **+NEW** at the bottom of the Machine Learning Studio (classic) window.



    b. Select **EXPERIMENT** > **Blank Experiment**.
    c. The experiment is given a default name that you can see at the top of the canvas. Select this text and rename it to A**utomobile Price Prediction.**
    d. To the left of the experiment canvas is a palette of datasets and modules. Type **automobile** in the Search box at the top of this palette to find the dataset labeled **Automobile price data (Raw)**. Drag this dataset to the experiment canvas.



    e. Hover on the 1st nod, right-click the mouse and visualize the dataset:



    f. In this dataset, each row represents an automobile, and the variables

associated with each automobile appear as columns. We'll predict the price in far-right column (column 26, titled "price") using the variables for a specific automobile.

g. Observe the normalized-losses columns, many values are missing; a screen snapshot and add it to you documentation.
h. Take a screen snapshot of the set of the columns that ends with the Price column and add it to your documentation.
i. Close the visualization window by clicking the "**x**" in the upper-right corner.

A. **Prepare the data:** A dataset usually requires some preprocessing before it can be analyzed. You might have noticed the missing values present in the columns of various rows. These missing values need to be cleaned so the model can analyze the data correctly. We'll remove any rows that have missing values. Also, the **normalized-losses** column has a large proportion of missing values, so we'll exclude that column from the model altogether.

- First, we add a module that removes the **normalized-losses** column completely. Then we add another module that removes any row that has missing data.

1. Type **select columns** in the search box at the top of the module palette to find the Select Columns in Dataset module. Then drag it to the experiment canvas. This module allows us to select which columns of data we want to include or exclude in the model.
2. Connect the output port of the **Automobile price data (Raw)** dataset to the input port of the Select Columns in Dataset.
3. Click the Select Columns in Dataset module and click **Launch column selector** in the **Properties** pane.
    1. On the left, click **With rules**
    2. Under **Begin With**, click **All columns**. These rules direct Select Columns in Dataset to pass through all the columns (except those columns we're about to exclude).
    3. From the drop-downs, select **Exclude** and **column names**, and then click inside the text box. A list of columns is displayed. Select **normalized-losses**, and it's added to the text box.
    4. Click the check mark (OK) button to close the column selector (on the lower right).
    5. Now the properties pane for **Select Columns in Dataset** indicates that it will pass through all columns from the dataset except **normalized-losses**.

- You can add a comment to a module by double-clicking the module and entering text. This can help you see at a glance what the module is doing in your experiment. In this case double-click the **Select Columns in Dataset** module and type the comment "Exclude normalized losses."

6. Drag the Clean Missing Data module to the experiment canvas and connect it to the Select Columns in Dataset module. In the **Properties** pane, select **Remove entire row** under **Cleaning mode**. These options direct Clean Missing Data to clean the data by removing rows that have any missing values. Double-click the module and type the comment "Remove missing value rows."
7. Run the experiment by clicking **RUN** at the bottom of the page.

    - When the experiment has finished running, all the modules have a green check mark to indicate that they finished successfully. Notice also the **Finished running** status in the upper-right corner.
    - Now we have clean data. If you want to view the cleaned dataset, click the left output port of the Clean Missing Data module and select **Visualize**. Notice that the **normalized-losses** column is no longer included, and there are no missing values.

Now that the data is clean, we're ready to specify what features we're going to use in the predictive model.

B. Define Features:

In machine learning, *features* are individual measurable properties of something you're interested in. In our dataset, each row represents one automobile, and each column is a feature of that automobile.

Finding a good set of features for creating a predictive model requires experimentation and knowledge about the problem you want to solve. Some features are better for predicting the target than others. Some features have a strong correlation with other features and can be removed. For example, city-mpg and highway-mpg are closely related so we can keep one and remove the other without significantly affecting the prediction. Let's build a model that uses a subset of the features in our dataset. You can come back later and select different features, run the experiment again, and see if you get better results. But to start, let's try the following features:

- make,
- body-style,
- wheel-base,
- engine-size,
- horsepower,
- peak-rpm,
- highway-mpg,
- price

j. Drag another Select Columns in Dataset module to the experiment canvas. Connect the left output port of the Clean Missing Data module to the input of the Select Columns in Dataset module.
k. Double-click the module and type "Select features for prediction."
l. Click **Launch column selector** in the **Properties** pane.
m. Click **With rules**.
n. Under **Begin With**, click **No columns**. In the filter row, select **Include** and **column names** and select our list of column names in the text box. This filter directs the module to not pass through any columns (features) except the ones that we specify.
o. Click the check mark (OK) button.

This module produces a filtered dataset containing only the features we want to pass to the learning algorithm we'll use in the next step. Later, you can return and try again with a different selection of features.

C. Choose and apply an algorithm

- Now that the data is ready, constructing a predictive model consists of training and testing. We'll use our data to train the model, and then we'll test the model to see how closely it's able to predict prices.
- *Classification* and *regression* are two types of supervised machine learning algorithms. Classification predicts an answer from a defined set of categories, such as a color (red, blue, or green). Regression is used to predict a number.
- Because we want to predict price, which is a number, we'll use a regression algorithm. For this example, we'll use a *linear regression* model.
- We train the model by giving it a set of data that includes the price. The model scans the data and look for correlations between an automobile's features and its price. Then we'll test the model - we'll give it a set of features for automobiles we're familiar with and see how close the model comes to predicting the known price.
- We'll use our data for both training the model and testing it by splitting the data into separate training and testing datasets.
    1. Select and drag the Split Data module to the experiment canvas and connect it to the last Select Columns in Dataset module.
    2. Click the Split Data module to select it. Find the **Fraction of rows in the first output dataset** (in the **Properties** pane to the right of the canvas) and set it to 0.75. This way, we'll use 75 percent of the data to train the model, and hold back 25 percent for testing.

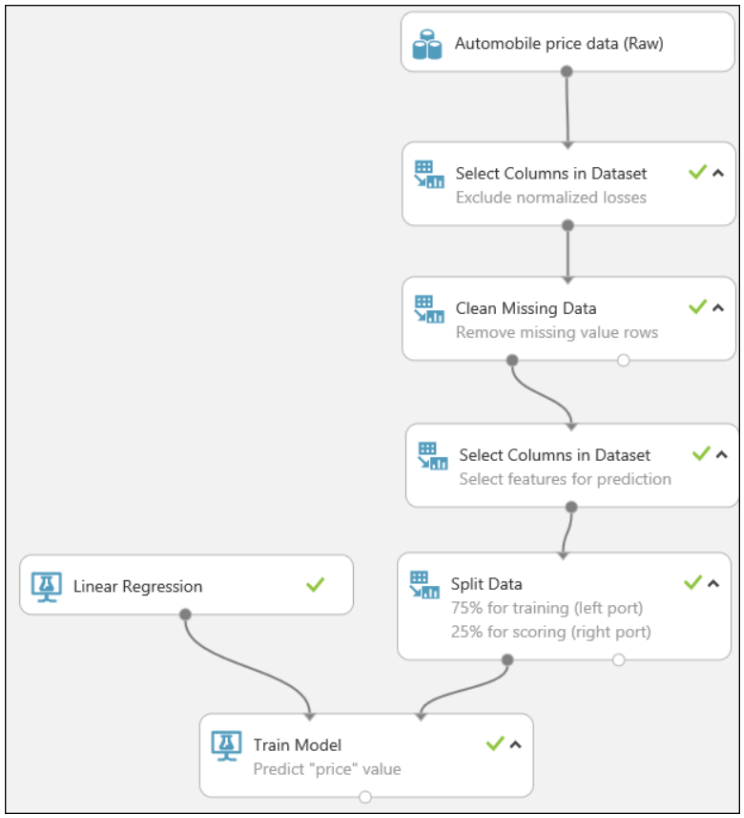
By changing the **Random seed** parameter, you can produce different random samples for training and testing. This parameter controls the seeding of the pseudo-random number generator.

3. Run the experiment. When the experiment is run, the Select Columns in Dataset and Split Data modules pass column definitions to the modules we'll be adding next.
4. To select the learning algorithm, expand the **Machine Learning** category in the module palette to the left of the canvas, and then expand **Initialize Model**. This displays several categories of modules that can be used to initialize machine learning algorithms. For this experiment, select the Linear Regression module under the **Regression** category, and drag it to the experiment canvas. (You can also find the module by typing "linear regression" in the palette Search box.)
5. Find and drag the Train Model module to the experiment canvas. Connect the output of the Linear Regression module to the left input of the Train Model module, and connect the training data output (left port) of the Split Data module to the right input of the Train Model module.
6. Click the Train Model module, click **Launch column selector** in the **Properties** pane, and then select the **price** column. **Price** is the value that our model is going to predict.

You select the **price** column in the column selector by moving it from the **Available columns** list to the **Selected columns** list.

7. Run the experiment.

We now have a trained regression model that can be used to score new automobile data to make price predictions.



D. Predict new automobile prices

Now that we've trained the model using 75 percent of our data, we can use it to score the other 25 percent of the data to see how well our model functions.

1. Find and drag the Score Model module to the experiment canvas. Connect the output of the Train Model module to the left input port of Score Model. Connect the test data output (right port) of the Split Data module to the right input port of Score Model.
2. Run the experiment and view the output from the Score Model module by clicking the output port of Score Model and select **Visualize**. The output shows the predicted values for price and the known values from the test data.
3. Finally, we test the quality of the results. Select and drag the Evaluate Model module to the experiment canvas, and connect the output of the Score Model module to the left input of Evaluate Model. The final experiment should look something like this:

   To view the output from the Evaluate Model module, click the output port, and then select **Visualize**.

4. Interpret the values of the of the statistic Coefficient of Determination.

Metrics

Mean Absolute Error	1656.147651
Root Mean Squared Error	2456.983209
Relative Absolute Error	0.276606
Relative Squared Error	0.089608
Coefficient of Determination	0.910392

5. Each activity/module should include “ECE 1779H” as a comment clause, as below