Stellar classification system:

1. Overview:

This project is the final nano-degree capstone project offered. This is the part of Udacity Nanodegree programme: Machine Learning Engineer for Microsoft Azure.

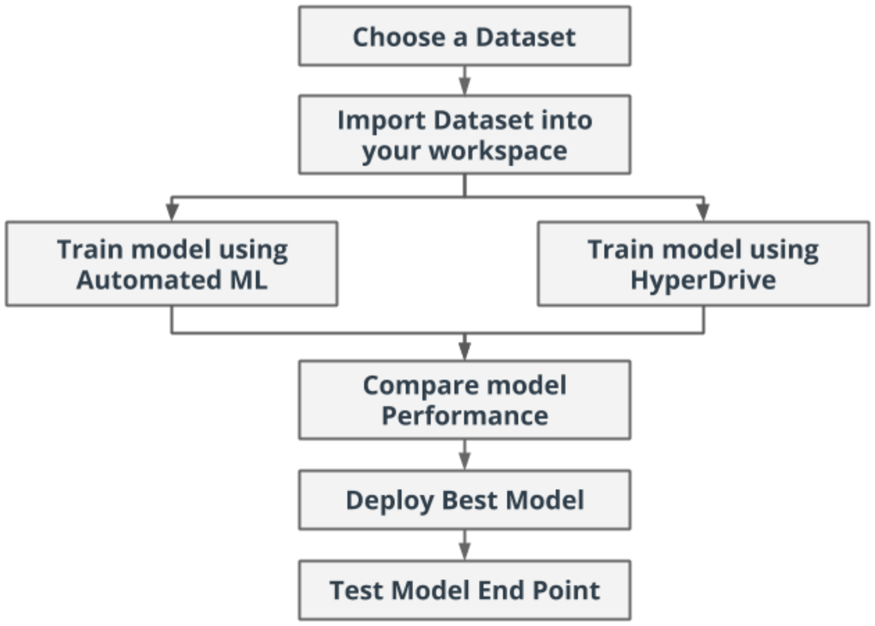
In this project I used an external dataset outside the Microsoft Azure environment to develop two Machine Learning experiments using Azure Auto ML and Hyperdrive.

Using hyper-drive we can choose hyper-parameters to optimize machine learning models.

The Azure Auto ML experiment and the Hyper Drive experiments were compared, the generated models are registered and the best performing model generated among the two was chosen and deployed as a web-service REST API using Azure Container Instance.

The overall task is explained in below figure:

2. Project Workflow diagram:


Choose a Dataset
Import Dataset into your workspace
Train model using Automated ML
Train model using HyperDrive
Compare model Performance
Deploy Best Model
Test Model End Point

3. Dataset:

Stellar Classification uses the spectral data of stars to categorize them into different categories. The modern stellar classification system is known as the Morgan–Keenan (MK) classification system. It uses the old HR classification system to categorize stars with their chromaticity and uses Roman numerals to categorize the star’s size. In this Dataset, we will be using Absolute Magnitude and B-V Color Index to Identify Giants and Dwarfs.

3.1. Data content overview:

This Dataset contains several features of Stars:

1. Vmag: Visual Apparent Magnitude of the Star
2. Plx: Distance Between the Star and the Earth
3. e_Plx: Standard Error of Plx (Drop the Row if you find the e_Plx is too high!)
4. B-V: B-V color index. (A hot star has a B-V color index close to 0 or negative, while a cool star has a B-V color index close to 2.0.
5. Amag: Absolute Magnitude of the Star

3.2. Task:

In this capstone project we will be predicting the TargetClass given the above feature as input.

3.3. Access:

The dataset has been uploaded into this github repository and it can be accessed using the link as below:

https://raw.githubusercontent.com/kalimi03/nd00333-capstone/master/Star3642_balanced.csv

Or it can also be downloadedfrom:

https://www.kaggle.com/vinesmsuic/star-categorization-giants-and-dwarfs/version/5

I have used method from_delimited_files('webURL') of the TabularDatasetFactory Class to retreive data from the csv file.

4. Automated ML:

Configuration	Description	Value
n_cross_validations	Total number of cross validation	5
experiment_timeout_minutes	This is used as an exit criteria, it defines how long, in minutes, the experiment should continue to run	25
max_concurrent_iterations	Represents the maximum number of iterations that would be executed in parallel	5
enable_early_stopping		True
primary_metric	The metric that Auto ML will optimize for model selection	Accuracy
compute_target	The compute target (computer cluster) to run the experiment on	compute_target
task	The type of task for example: 'classification', 'regression', or 'forecasting'.	Classification
training_data	Training data, contains both features and label columns	dataset
label_column_name	Target column name	TargetClass
path	Path to the project folder	project_folder

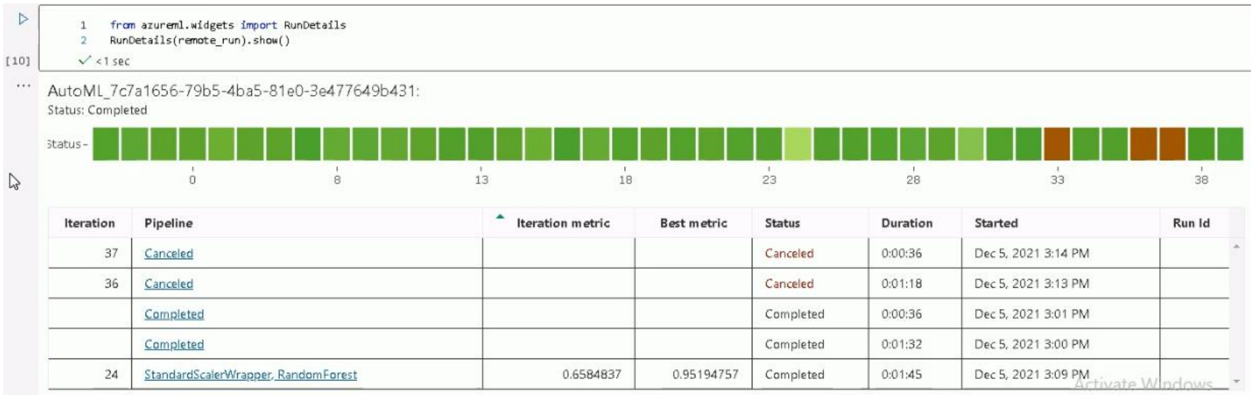
5. Results

In our experiment we found out that VotingEnsemble to be the best model based on the accuracy metric. The accuracy score for this model was 0.951.

The following would be done to improve the Auto ML run performance.

1. Increase the number of cross validation to improve model accuracy.
2. Experiment time out would not be specified so the Auto ML run can produce the best model at its own stipulated time.

Below are the screen shots created during auto-ml run. The best model for the experiment can be seen below:

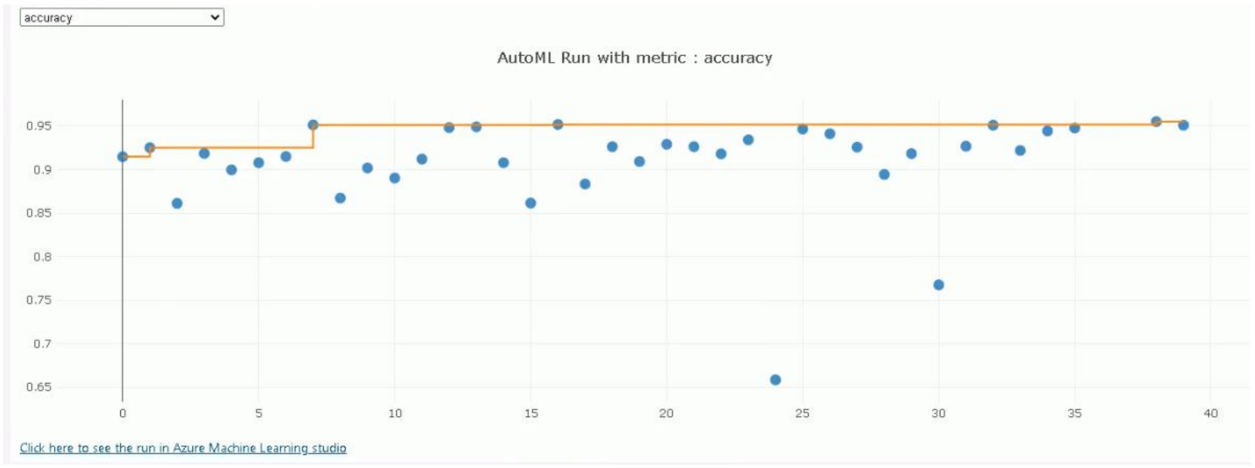


```
from azureml.widgets import RunDetails
RunDetails(remote_run).show()
```

[10] ✓ <1 sec

AutoML_7c7a1656-79b5-4ba5-81e0-3e477649b431:
Status: Completed

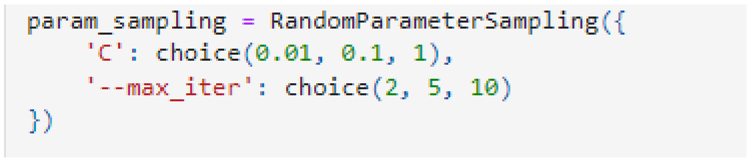
Iteration	Pipeline	Iteration metric	Best metric	Status	Duration	Started	Run Id
37	Canceled			Canceled	0:00:36	Dec 5, 2021 3:14 PM	
36	Canceled			Canceled	0:01:18	Dec 5, 2021 3:13 PM	
	Completed			Completed	0:00:36	Dec 5, 2021 3:01 PM	
	Completed			Completed	0:01:32	Dec 5, 2021 3:00 PM	
24	StandardScalerWrapper, RandomForest	0.6584837	0.95194757	Completed	0:01:45	Dec 5, 2021 3:09 PM	



6. Hyper-parameter Tuning:

I have selected Logistic Regression from Scikit-Learn library for this experiment as it is easier to implement, interpret, and efficient to train models.

The parameter sampling I used in this experiment was Random Parameter sampling. In this experiment I optimized two most important hyperparameter values for logistic regression by defining two hyperparameters search space as it can be seen in the python code below:

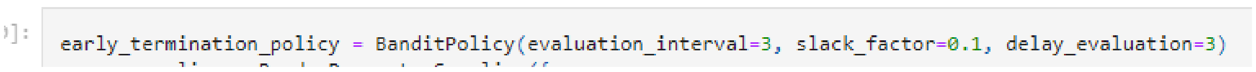


```
param_sampling = RandomParameterSampling({
    'C': choice(0.01, 0.1, 1),
    '--max_iter': choice(2, 5, 10)
})
```

--C with a choice of discrete values (0.01, 0.1, 1) is the Inverse Regularization strength which helps to reduce overfitting. The smaller values cause stronger regularization.

--max_iter with a choice of discrete values (2, 5, 10), is the maximum number of iterations to converge. This convergence maximizes the model's accuracy.

Another hyper-parameter is BanditPolicy: Bandit is an early termination policy based on slack factor/slack amount and evaluation interval. The policy early terminates any runs where the primary metric is not within the specified slack factor/slack amount with respect to the best performing training run. Below is the python lie of code for defining this hyper-parameter:



```
]:  early_termination_policy = BanditPolicy(evaluation_interval=3, slack_factor=0.1, delay_evaluation=3)
```

7. Hyperdrive Results

The best model from the Hyperdrive run generates an accuracy of 0.8979.

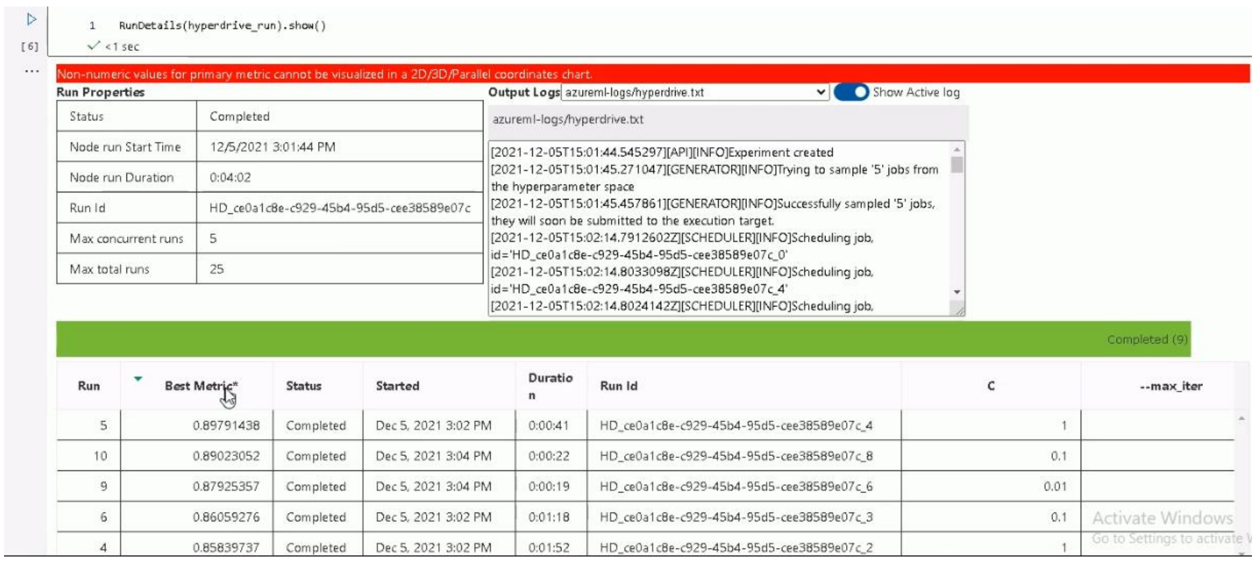
7.1. Model Improvement

To improve the model:

- Grid sampling would be used in in place of Random sampling can be chosen. Grid sampling may provide a little bit of performance as it searches over all possible values.

- The parameters selected (C and max_iter) are limited in range. An experiment can be done with better range of value. But It may take more time and compute power to process.

Below is the screenshots of RunDetails of the hyperdrive run;



```
RunDetails(hyperdrive_run).show()
[6]  ✓ <1 sec
```

Non-numeric values for primary metric cannot be visualized in a 2D/3D/Parallel coordinates chart.

Run Properties

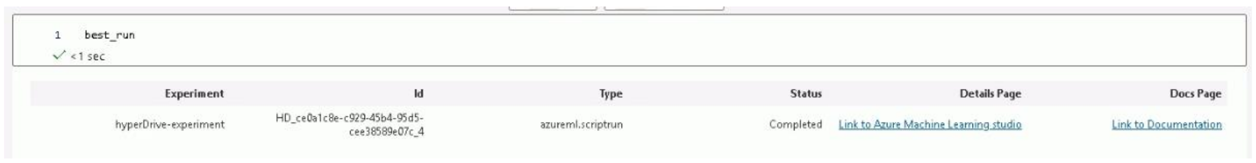
Status	Completed
Node run Start Time	12/5/2021 3:01:44 PM
Node run Duration	0:04:02
Run Id	HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c
Max concurrent runs	5
Max total runs	25

Output Logs azureml-logs/hyperdrive.txt — Show Active log

```
azureml-logs/hyperdrive.txt

[2021-12-05T15:01:44.545297][API][INFO]Experiment created
[2021-12-05T15:01:45.271047][GENERATOR][INFO]Trying to sample '5' jobs from
the hyperparameter space
[2021-12-05T15:01:45.457861][GENERATOR][INFO]Successfully sampled '5' jobs,
they will soon be submitted to the execution target.
[2021-12-05T15:02:14.7912602Z][SCHEDULER][INFO]Scheduling job,
id='HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_0'
[2021-12-05T15:02:14.8033098Z][SCHEDULER][INFO]Scheduling job,
id='HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_4'
[2021-12-05T15:02:14.8024142Z][SCHEDULER][INFO]Scheduling job,
```

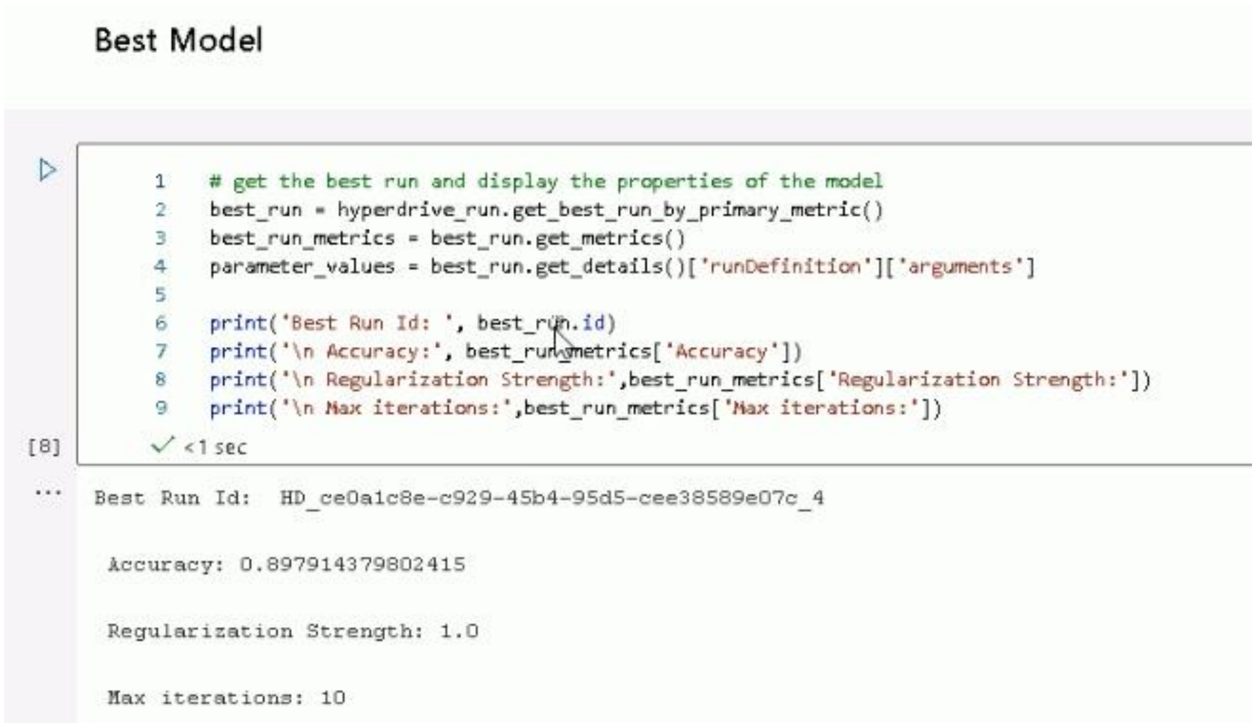
Completed (9)

Run	Best Metric*	Status	Started	Duration	Run Id	C	--max_iter
5	0.89791438	Completed	Dec 5, 2021 3:02 PM	0:00:41	HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_4	1	
10	0.89023052	Completed	Dec 5, 2021 3:04 PM	0:00:22	HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_8	0.1	
9	0.87925357	Completed	Dec 5, 2021 3:04 PM	0:00:19	HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_6	0.01	
6	0.86059276	Completed	Dec 5, 2021 3:02 PM	0:01:18	HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_3	0.1	
4	0.85839737	Completed	Dec 5, 2021 3:02 PM	0:01:52	HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_2	1	



```
best_run
✓ <1 sec
```

Experiment	Id	Type	Status	Details Page	Docs Page
hyperDrive-experiment	HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_4	azureml.scriptrun	Completed	Link to Azure Machine Learning studio	Link to Documentation



Best Model

```
# get the best run and display the properties of the model
best_run = hyperdrive_run.get_best_run_by_primary_metric()
best_run_metrics = best_run.get_metrics()
parameter_values = best_run.get_details()['runDefinition']['arguments']

print('Best Run Id: ', best_run.id)
print('\n Accuracy:', best_run_metrics['Accuracy'])
print('\n Regularization Strength:',best_run_metrics['Regularization Strength:'])
print('\n Max iterations:',best_run_metrics['Max iterations:'])
```

```
[8]  ✓ <1 sec

Best Run Id:  HD_ce0a1c8e-c929-45b4-95d5-cee38589e07c_4

 Accuracy: 0.897914379802415

 Regularization Strength: 1.0

 Max iterations: 10
```

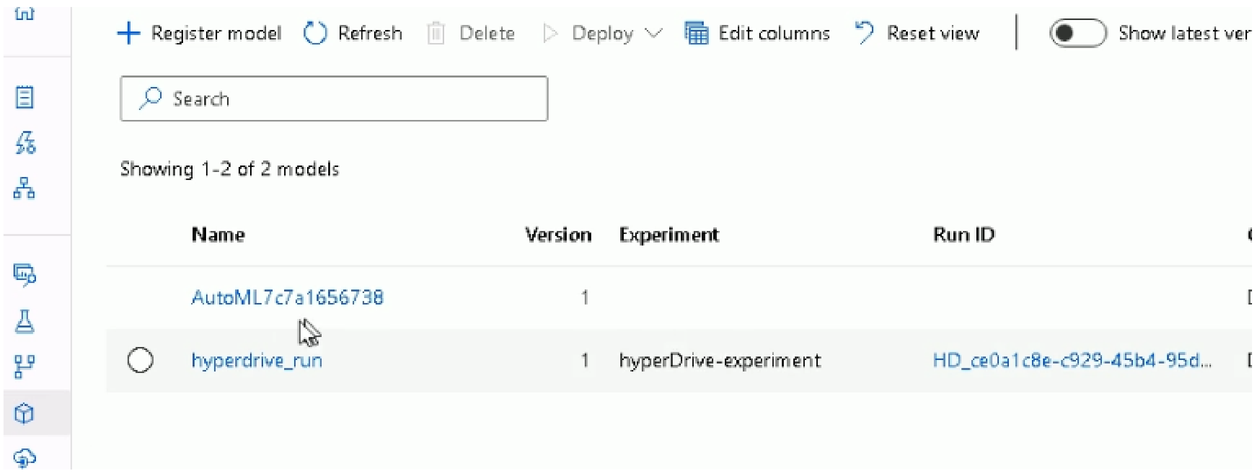
8. Model Deployment

The best model VotingEnsemble which was an AutoML model was deployed in this project, since it produced the best accuracy of 0.95 as compared to the hyperdrive run with an accuracy of 0.89.

The best model model.pkl file from the Auto ML run was first retrieved together with its conda environment script conda_env.yml and scoring script score.py.

Below are the steps for model deployment:

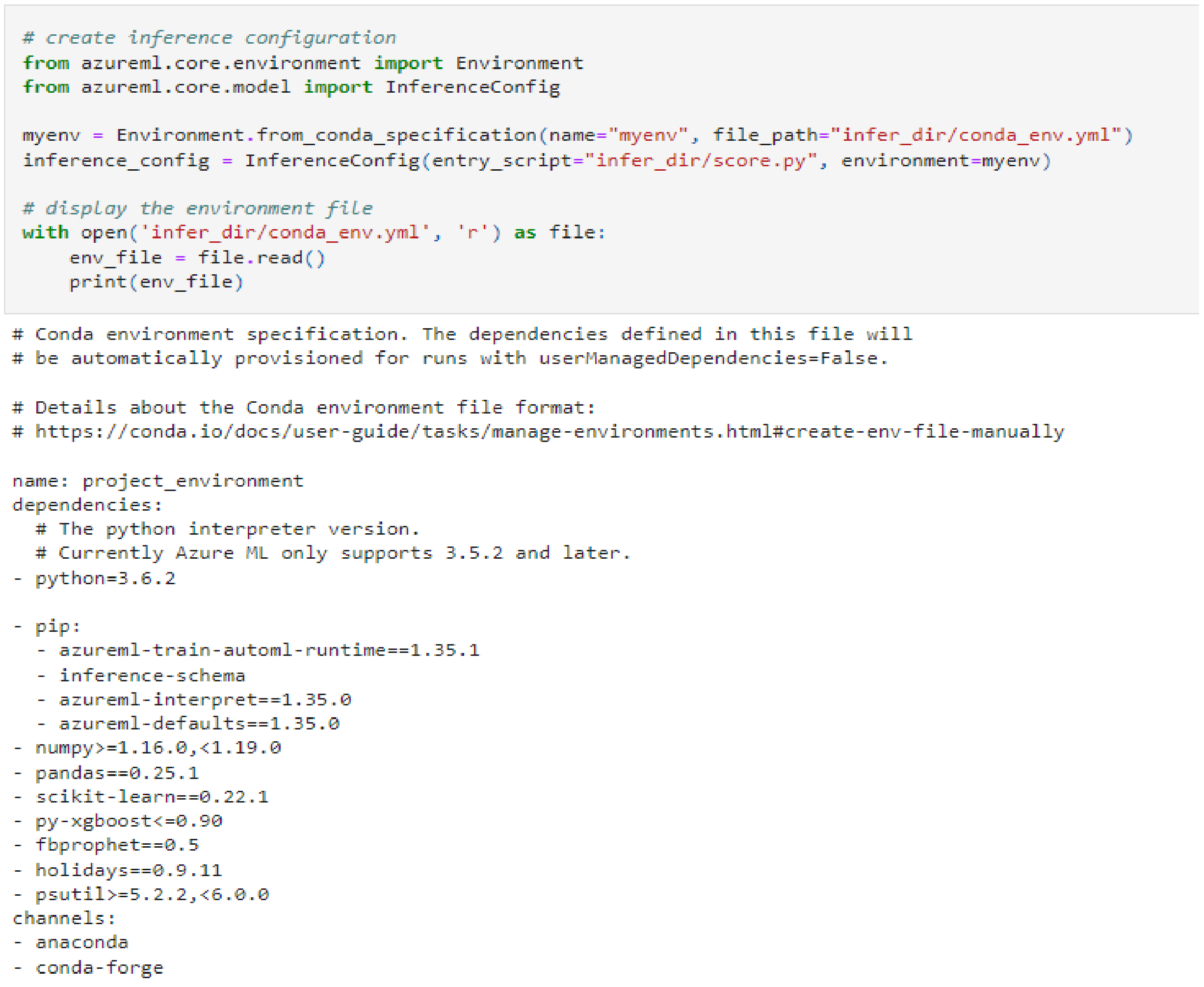
8.1. The model once trained will be registered (Azure ML model). As shown below:



Showing 1-2 of 2 models

Name	Version	Experiment	Run ID
AutoML7c7a1656738	1		
hyperdrive_run	1	hyperDrive-experiment	HD_ce0a1c8e-c929-45b4-95d...

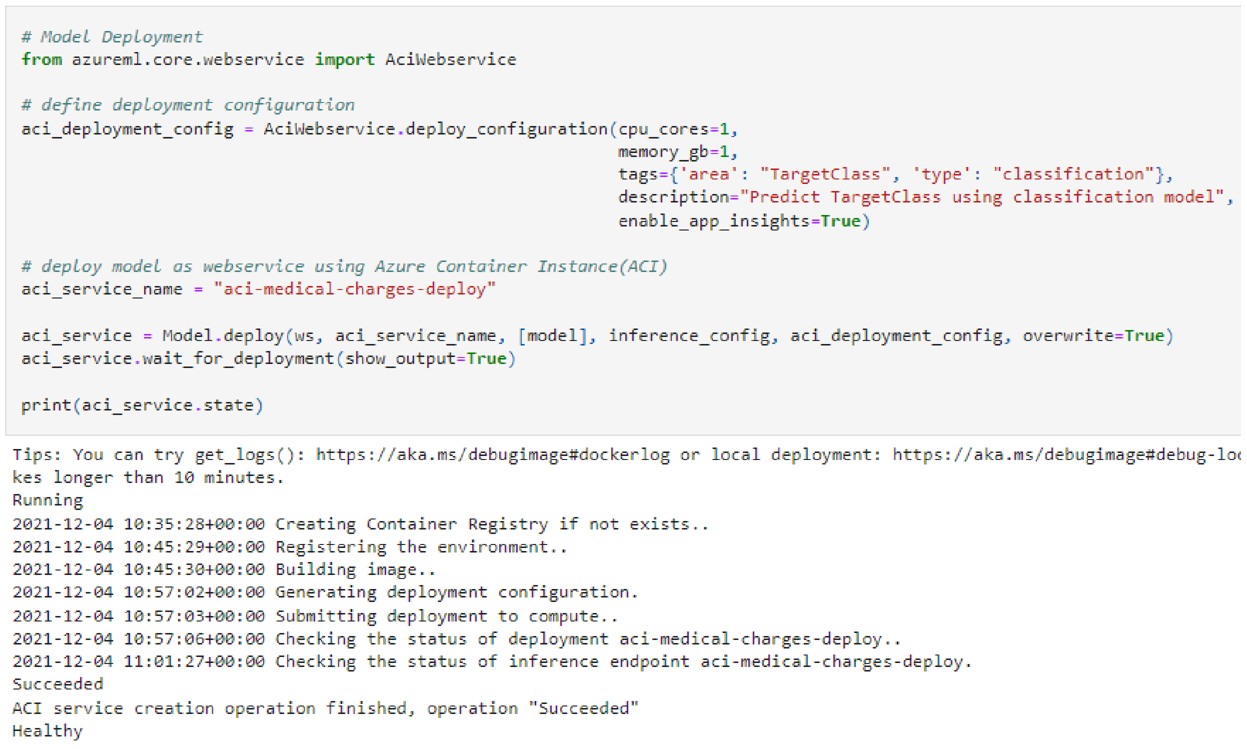
8.2. An inference configuration was created from the downloaded conda_env.yml and score.py to make sure that the software dependencies and resources needed for deployment is intact. Below figure shows conda environment specifications:



```python
# create inference configuration
from azureml.core.environment import Environment
from azureml.core.model import InferenceConfig

myenv = Environment.from_conda_specification(name="myenv", file_path="infer_dir/conda_env.yml")
inference_config = InferenceConfig(entry_script="infer_dir/score.py", environment=myenv)

# display the environment file
with open('infer_dir/conda_env.yml', 'r') as file:
    env_file = file.read()
    print(env_file)
```

```
# Conda environment specification. The dependencies defined in this file will
# be automatically provisioned for runs with userManagedDependencies=False.

# Details about the Conda environment file format:
# https://conda.io/docs/user-guide/tasks/manage-environments.html#create-env-file-manually

name: project_environment
dependencies:
  # The python interpreter version.
  # Currently Azure ML only supports 3.5.2 and later.
- python=3.6.2

- pip:
  - azureml-train-automl-runtime==1.35.1
  - inference-schema
  - azureml-interpret==1.35.0
  - azureml-defaults==1.35.0
- numpy>=1.16.0,<1.19.0
- pandas==0.25.1
- scikit-learn==0.22.1
- py-xgboost<=0.90
- fbprophet==0.5
- holidays==0.9.11
- psutil>=5.2.2,<6.0.0
channels:
- anaconda
- conda-forge
```

8.3. The model was then deployed with the inference configuration as an Azure Container Instance (ACI) webservice.



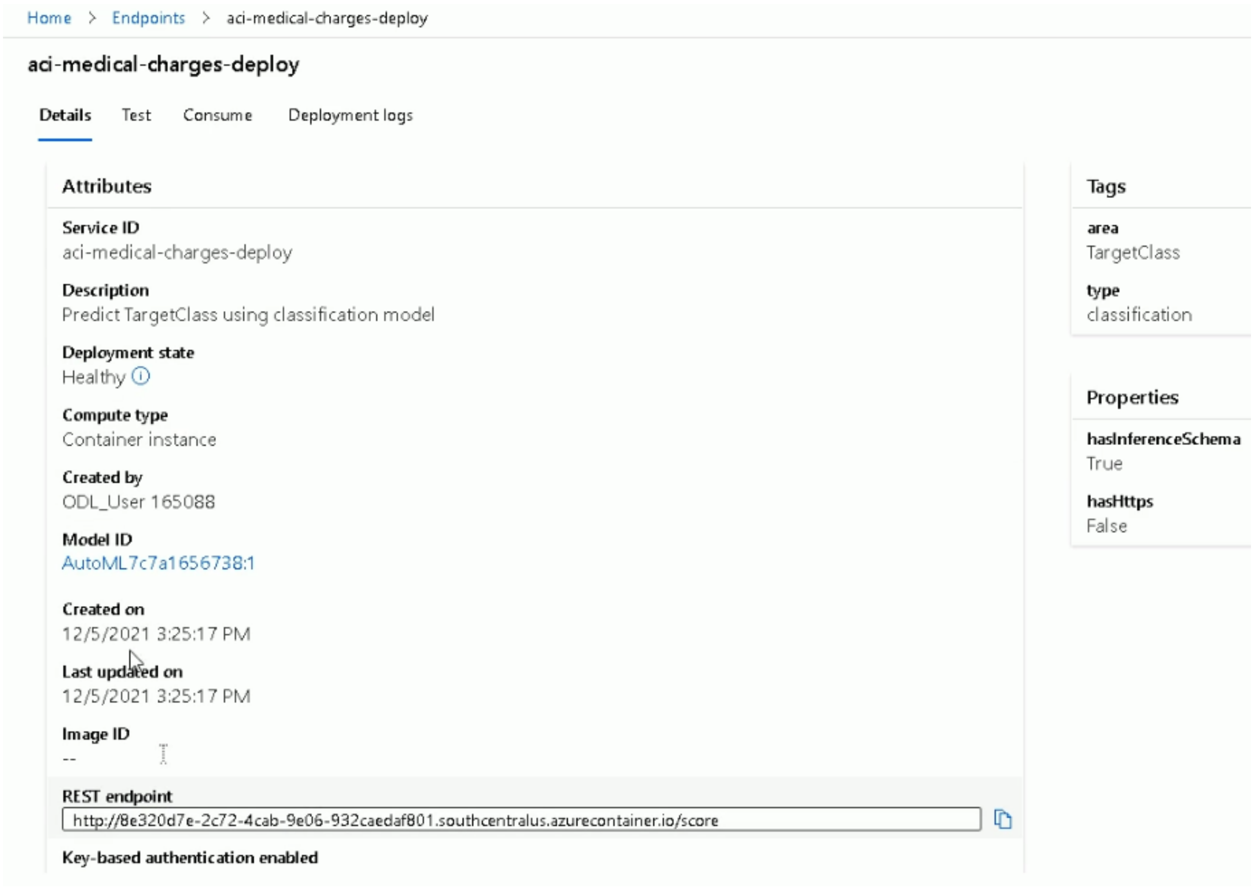
```
# Model Deployment
from azureml.core.webservice import AciWebservice

# define deployment configuration
aci_deployment_config = AciWebservice.deploy_configuration(cpu_cores=1,
                                                           memory_gb=1,
                                                           tags={'area': "TargetClass", 'type': "classification"},
                                                           description="Predict TargetClass using classification model",
                                                           enable_app_insights=True)

# deploy model as webservice using Azure Container Instance(ACI)
aci_service_name = "aci-medical-charges-deploy"

aci_service = Model.deploy(ws, aci_service_name, [model], inference_config, aci_deployment_config, overwrite=True)
aci_service.wait_for_deployment(show_output=True)

print(aci_service.state)
```

```
Tips: You can try get_logs(): https://aka.ms/debugimage#dockerlog or local deployment: https://aka.ms/debugimage#debug-loc
kes longer than 10 minutes.
Running
2021-12-04 10:35:28+00:00 Creating Container Registry if not exists..
2021-12-04 10:45:29+00:00 Registering the environment..
2021-12-04 10:45:30+00:00 Building image..
2021-12-04 10:57:02+00:00 Generating deployment configuration.
2021-12-04 10:57:03+00:00 Submitting deployment to compute..
2021-12-04 10:57:06+00:00 Checking the status of deployment aci-medical-charges-deploy..
2021-12-04 11:01:27+00:00 Checking the status of inference endpoint aci-medical-charges-deploy.
Succeeded
ACI service creation operation finished, operation "Succeeded"
Healthy
```

9. Model endpoint

The screenshot below shows the healthy status of the deployed model endpoint which indicates

That the model deployed is active:



aci-medical-charges-deploy

Attributes

Service ID
aci-medical-charges-deploy

Description
Predict TargetClass using classification model

Deployment state
Healthy

Compute type
Container instance

Created by
ODL_User 165088

Model ID
AutoML7c7a1656738:1

Created on
12/5/2021 3:25:17 PM

Last updated on
12/5/2021 3:25:17 PM

Image ID
--

REST endpoint
http://8e320d7e-2c72-4cab-9e06-932caedaf801.southcentralus.azurecontainer.io/score

Key-based authentication enabled

Tags

area
TargetClass

type
classification

Properties

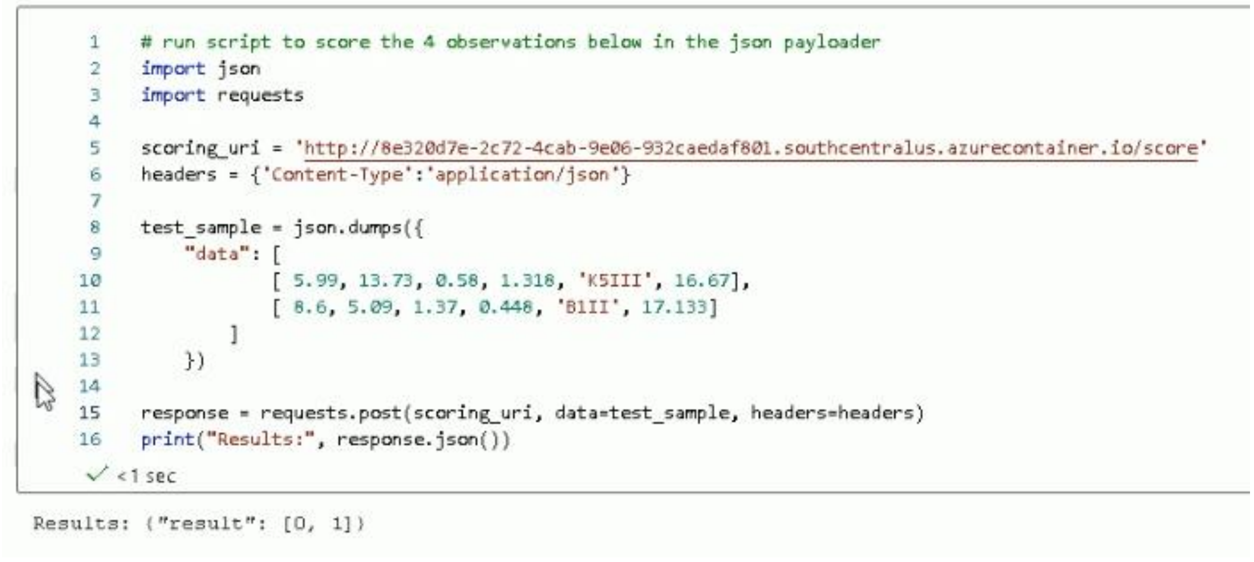
hasInferenceSchema
True

hasHttps
False

Consume deployed model with sample:

1. To consume the deployed model the http endpoint url was embeded into a python code with sample data observations loaded into a json payloader.
2. HTTP webservice POST request is then sent to the enpoint url and the response from the url is captured as a json data.
3. Any application or service is capable of consuming the model using the enpoint url by making http request calls to the webservice.

10. Run demo with sample data:

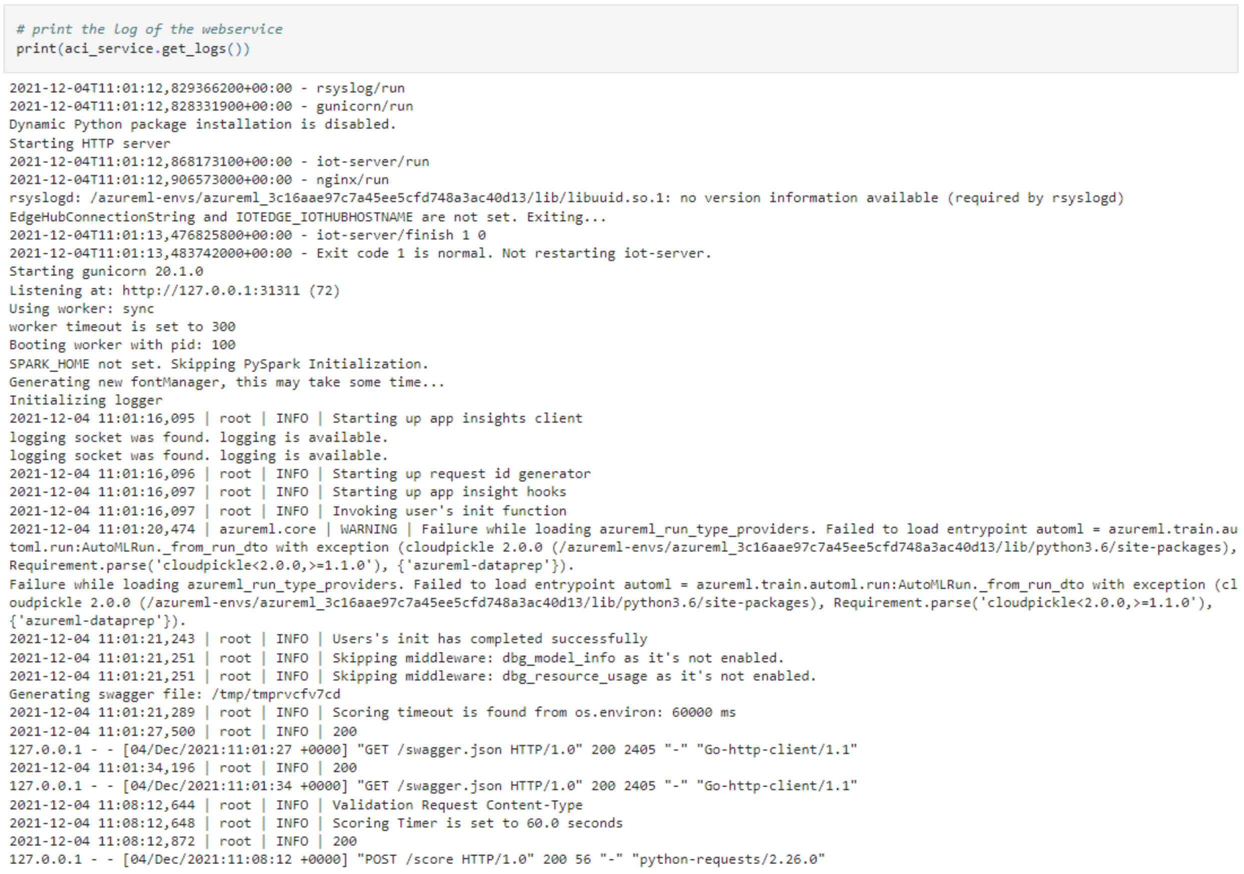


```python
# run script to score the 4 observations below in the json payloader
import json
import requests

scoring_uri = 'http://8e320d7e-2c72-4cab-9e06-932caedaf801.southcentralus.azurecontainer.io/score'
headers = {'Content-Type':'application/json'}

test_sample = json.dumps({
    "data": [
            [ 5.99, 13.73, 0.58, 1.318, 'K5III', 16.67],
            [ 8.6, 5.09, 1.37, 0.448, 'B1II', 17.133]
        ]
    })

response = requests.post(scoring_uri, data=test_sample, headers=headers)
print("Results:", response.json())
```

✓ <1 sec

```
Results: {"result": [0, 1]}
```

The logs of the web service are as follows:



```python
# print the log of the webservice
print(aci_service.get_logs())
```

```
2021-12-04T11:01:12,829366200+00:00 - rsyslog/run
2021-12-04T11:01:12,828331900+00:00 - gunicorn/run
Dynamic Python package installation is disabled.
Starting HTTP server
2021-12-04T11:01:12,868173100+00:00 - iot-server/run
2021-12-04T11:01:12,906573000+00:00 - nginx/run
rsyslogd: /azureml-envs/azureml_3c16aae97c7a45ee5cfd748a3ac40d13/lib/libuuid.so.1: no version information available (required by rsyslogd)
EdgeHubConnectionString and IOTEDGE_IOTHUBHOSTNAME are not set. Exiting...
2021-12-04T11:01:13,476825800+00:00 - iot-server/finish 1 0
2021-12-04T11:01:13,483742000+00:00 - Exit code 1 is normal. Not restarting iot-server.
Starting gunicorn 20.1.0
Listening at: http://127.0.0.1:31311 (72)
Using worker: sync
worker timeout is set to 300
Booting worker with pid: 100
SPARK_HOME not set. Skipping PySpark Initialization.
Generating new fontManager, this may take some time...
Initializing logger
2021-12-04 11:01:16,095 | root | INFO | Starting up app insights client
logging socket was found. logging is available.
logging socket was found. logging is available.
2021-12-04 11:01:16,096 | root | INFO | Starting up request id generator
2021-12-04 11:01:16,097 | root | INFO | Starting up app insight hooks
2021-12-04 11:01:16,097 | root | INFO | Invoking user's init function
2021-12-04 11:01:20,474 | azureml.core | WARNING | Failure while loading azureml_run_type_providers. Failed to load entrypoint automl = azureml.train.au
toml.run:AutoMLRun._from_run_dto with exception (cloudpickle 2.0.0 (/azureml-envs/azureml_3c16aae97c7a45ee5cfd748a3ac40d13/lib/python3.6/site-packages),
Requirement.parse('cloudpickle<2.0.0,>=1.1.0'), {'azureml-dataprep'}).
Failure while loading azureml_run_type_providers. Failed to load entrypoint automl = azureml.train.automl.run:AutoMLRun._from_run_dto with exception (cl
oudpickle 2.0.0 (/azureml-envs/azureml_3c16aae97c7a45ee5cfd748a3ac40d13/lib/python3.6/site-packages), Requirement.parse('cloudpickle<2.0.0,>=1.1.0'),
{'azureml-dataprep'}).
2021-12-04 11:01:21,243 | root | INFO | Users's init has completed successfully
2021-12-04 11:01:21,251 | root | INFO | Skipping middleware: dbg_model_info as it's not enabled.
2021-12-04 11:01:21,251 | root | INFO | Skipping middleware: dbg_resource_usage as it's not enabled.
Generating swagger file: /tmp/tmprvcfv7cd
2021-12-04 11:01:21,289 | root | INFO | Scoring timeout is found from os.environ: 60000 ms
2021-12-04 11:01:27,500 | root | INFO | 200
127.0.0.1 - - [04/Dec/2021:11:01:27 +0000] "GET /swagger.json HTTP/1.0" 200 2405 "-" "Go-http-client/1.1"
2021-12-04 11:01:34,196 | root | INFO | 200
127.0.0.1 - - [04/Dec/2021:11:01:34 +0000] "GET /swagger.json HTTP/1.0" 200 2405 "-" "Go-http-client/1.1"
2021-12-04 11:08:12,644 | root | INFO | Validation Request Content-Type
2021-12-04 11:08:12,648 | root | INFO | Scoring Timer is set to 60.0 seconds
2021-12-04 11:08:12,872 | root | INFO | 200
127.0.0.1 - - [04/Dec/2021:11:08:12 +0000] "POST /score HTTP/1.0" 200 56 "-" "python-requests/2.26.0"
```

11. Screen cast video link:

https://youtu.be/01WuPF0RhcQ