**Problem Statement**

To build a classification methodology to predict the type of Mushroom and whether it is edible or poisnous based on the given training data.

**Architecture**



**Data Description**

Data Description: This dataset describes mushrooms in terms of their physical characteristics. They are classified into: poisonous or edible.

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s

2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s

3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y

4. bruises?: bruises=t,no=f

5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s

6. gill-attachment: attached=a,descending=d,free=f,notched=n

7. gill-spacing: close=c,crowded=w,distant=d

8. gill-size: broad=b,narrow=n

9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y

10. stalk-shape: enlarging=e,tapering=t

11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?

12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s

13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s

14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y

15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y

16. veil-type: partial=p,universal=u

17. veil-color: brown=n,orange=o,white=w,yellow=y

18. ring-number: none=n,one=o,two=t

19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z

20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y

21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y

22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

Apart from training files, we also require a "schema" file from the client, which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns, and their datatype.

**Data Validation**

In this step, we perform different sets of validation on the given set of training files.

1. Name Validation- We validate the name of the files based on the given name in the schema file. We have created a regex pattern as per the name given in the schema file to use for validation. After validating the pattern in the name, we check for the length of date in the file name as well as the length of time in the file name. If all the values are as per requirement, we move such files to "Good\_Data\_Folder" else we move such files to "Bad\_Data\_Folder."
2. Number of Columns - We validate the number of columns present in the files, and if it doesn't match with the value given in the schema file, then the file is moved to "Bad\_Data\_Folder."
3. Name of Columns - The name of the columns is validated and should be the same as given in the schema file. If not, then the file is moved to "Bad\_Data\_Folder".
4. The datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If the datatype is wrong, then the file is moved to "Bad\_Data\_Folder".
5. Null values in columns - If any of the columns in a file have all the values as NULL or missing, we discard such a file and move it to "Bad\_Data\_Folder".

**Data Insertion in Database**

1) Database Creation and connection - Create a database with the given name passed. If the database is already created, open the connection to the database.

2) Table creation in the database - Table with name - "Good\_Data", is created in the database for inserting the files in the "Good\_Data\_Folder" based on given column names and datatype in the schema file. If the table is already present, then the new table is not created and new files are inserted in the already present table as we want training to be done on new as well as old training files.

3) Insertion of files in the table - All the files in the "Good\_Data\_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad\_Data\_Folder".

**Model Training**

1) Data Export from Db - The data in a stored database is exported as a CSV file to be used for model training.

2) Data Preprocessing

a) Drop columns not useful for training the model. Such columns were selected while doing the EDA.

b) Replace the invalid values with numpy “nan” so we can use imputer on such values.

c) Encode the categorical values

d) Check for null values in the columns. If present, impute the null values using the Single imputer.

3) Clustering - KMeans algorithm is used to create clusters in the preprocessed data. The optimum number of clusters is selected by plotting the elbow plot, and for the dynamic selection of the number of clusters, we are using "KneeLocator" function. The idea behind clustering is to implement different algorithms

To train data in different clusters. The Kmeans model is trained over preprocessed data and the model is saved for further use in prediction.

4) Model Selection - After clusters are created, we find the best model for each cluster. We are using two algorithms, "Random Forest" and "KNN". For each cluster, both the algorithms are passed with the best parameters derived from GridSearch. We calculate the AUC scores for both models and select the model with the best score. Similarly, the model is selected for each cluster. All the models for every cluster are saved for use in prediction.

**Prediction Data Description**

Client will send the data in multiple set of files in batches at a given location. Data will contain Mushroom properties in 22 columns.

Apart from prediction files, we also require a "schema" file from client which contains all the relevant information about the training files such as:

Name of the files, Length of Date value in FileName, Length of Time value in FileName, Number of Columns, Name of the Columns and their datatype.

**Data Validation**

In this step, we perform different sets of validation on the given set of training files.

1) Name Validation- We validate the name of the files on the basis of given Name in the schema file. We have created a regex pattern as per the name given in schema file, to use for validation. After validating the pattern in the name, we check for length of date in the file name as well as length of time in the file name. If all the values are as per requirement, we move such files to "Good\_Data\_Folder" else we move such files to "Bad\_Data\_Folder".

2) Number of Columns - We validate the number of columns present in the files, if it doesn't match with the value given in the schema file then the file is moved to "Bad\_Data\_Folder".

3) Name of Columns - The name of the columns is validated and should be same as given in the schema file. If not, then the file is moved to "Bad\_Data\_Folder".

4) Datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If dataype is wrong then the file is moved to "Bad\_Data\_Folder".

5) Null values in columns - If any of the columns in a file has all the values as NULL or missing, we discard such file and move it to "Bad\_Data\_Folder".

**Data Insertion in Database**

1) Database Creation and connection - Create database with the given name passed. If the database is already created, open the connection to the database.

2) Table creation in the database - Table with name - "Good\_Data", is created in the database for inserting the files in the "Good\_Data\_Folder" on the basis of given column names and datatype in the schema file. If table is already present then new table is not created, and new files are inserted the already present table as we want training to be done on new as well old training files.

3) Insertion of files in the table - All the files in the "Good\_Data\_Folder" are inserted in the above-created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "Bad\_Data\_Folder".

**Prediction**

1) Data Export from Db - The data in the stored database is exported as a CSV file to be used for prediction.

2) Data Preprocessing

a) Drop columns not useful for training the model. Such columns were selected while doing the EDA.

b) Replace the invalid values with numpy “nan” so we can use imputer on such values.

c) Encode the categorical values

d) Check for null values in the columns. If present, impute the null values using the Single imputer.

3) Clustering - KMeans model created during training is loaded, and clusters for the preprocessed prediction data is predicted.

4) Prediction - Based on the cluster number, the respective model is loaded and is used to predict the data for that cluster.

5) Once the prediction is made for all the clusters, the predictions along with the original names before label encoder are saved in a CSV file at a given location and the location is returned to the client.

**Deployment**

We will be deploying the model to the Pivotal Web Services Platform.

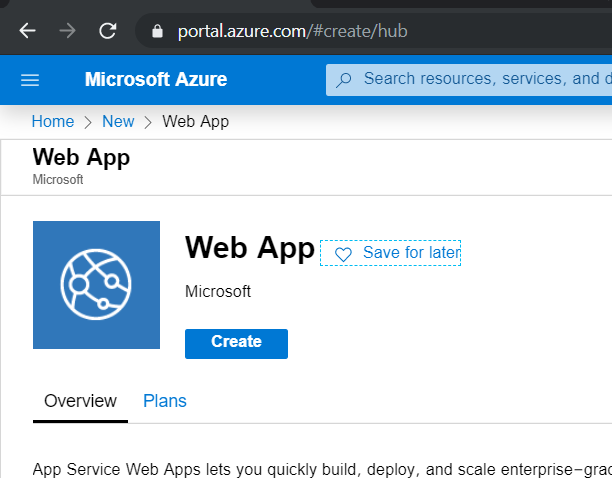
This is a workflow diagram for the prediction of using the trained model.

**Now let’s see the Mushroom classification project folder structure.**

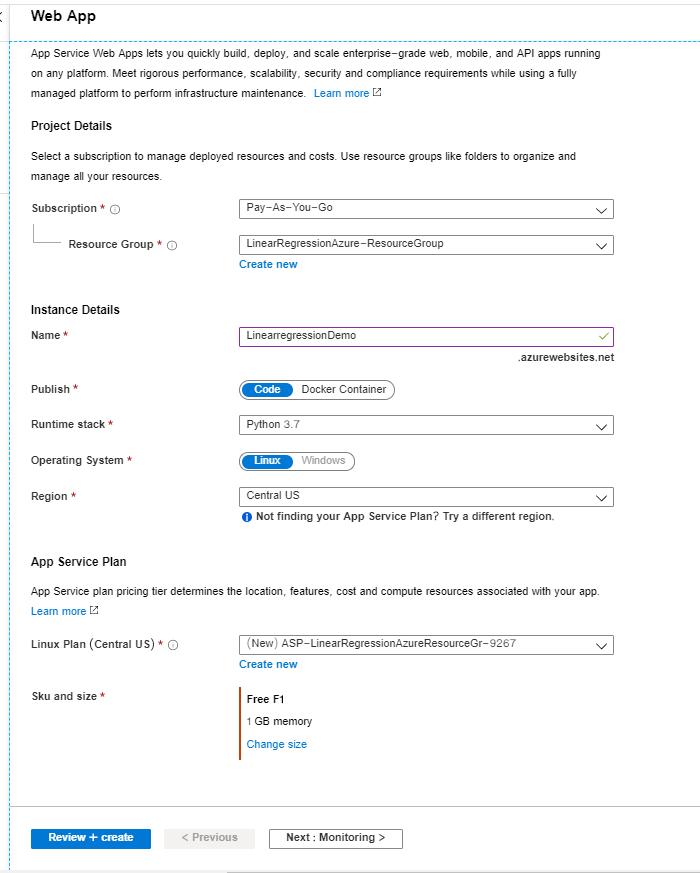
**requirements.txt** file consists of all the packages that you need to deploy the app in the cloud.

1. **Deployment to Azure:**

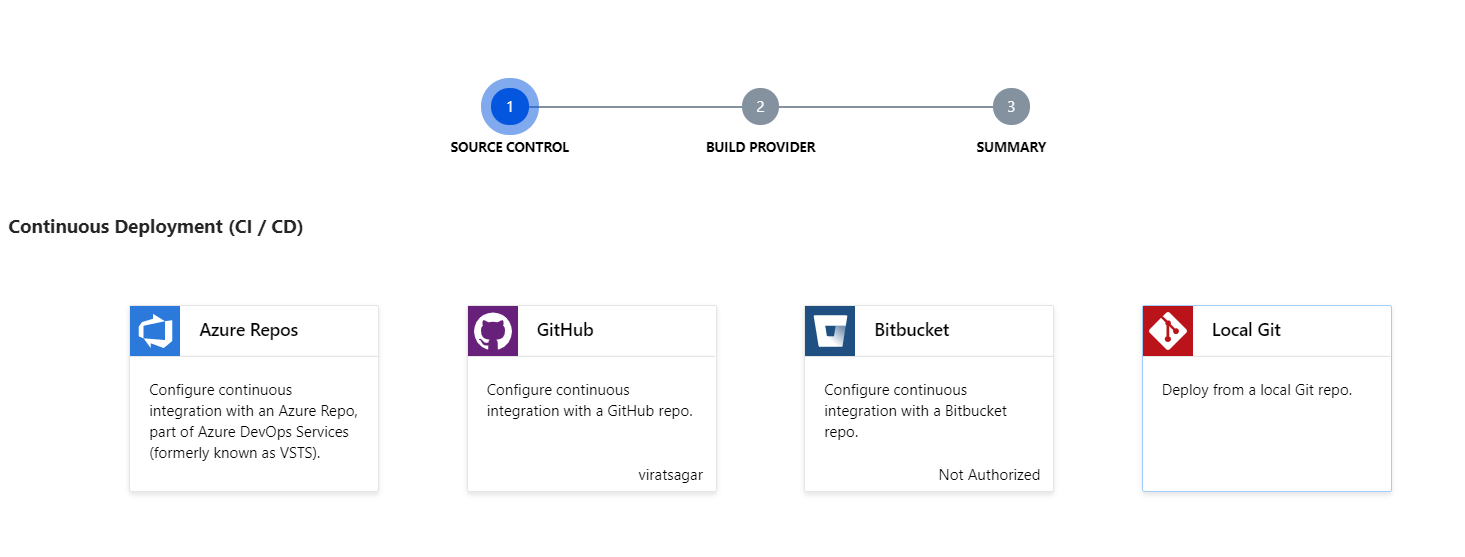
* Go to <https://portal.azure.com/> and create an account if already haven’t created one.
* Go to the Azure account and create a web app.



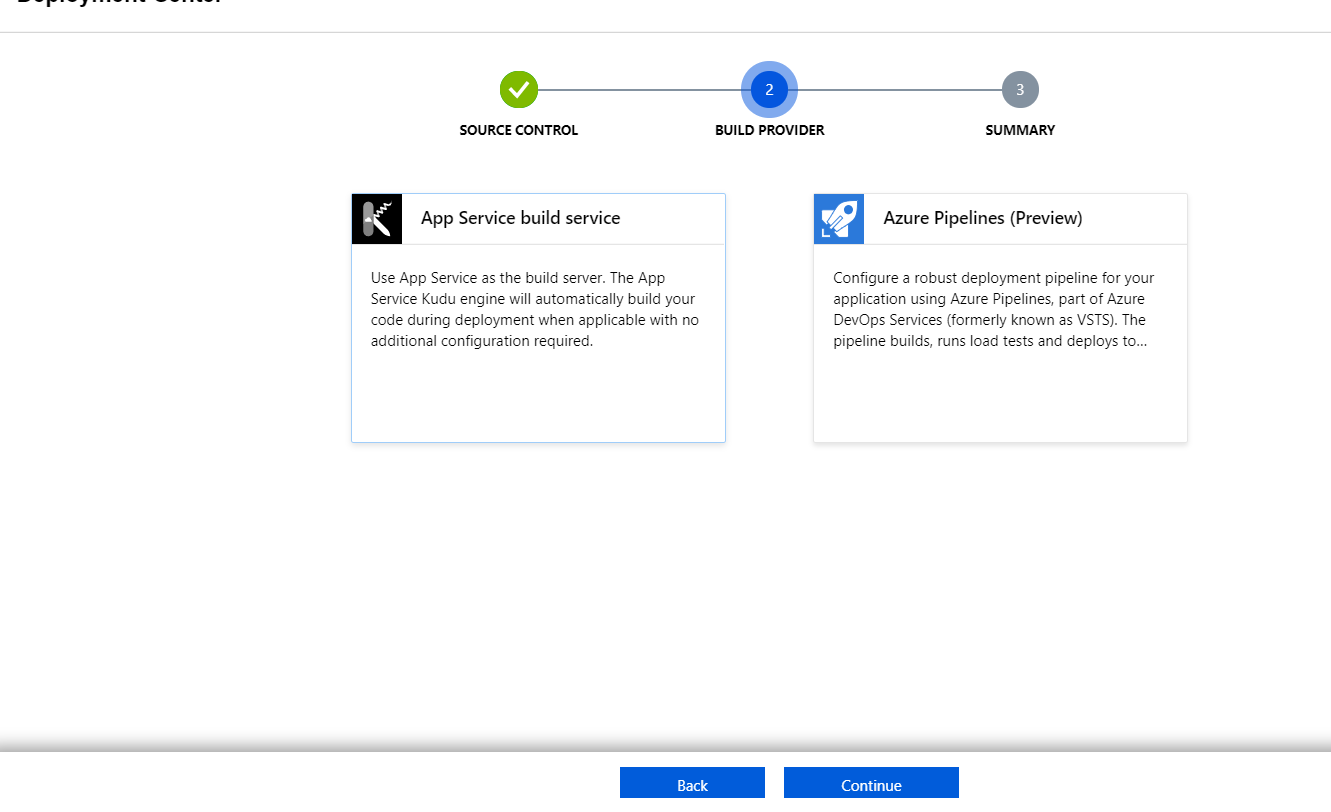
* Provide the app name, resource group(create new if necessary), runtime stack(Python 3.7), region, select the 1 GB size, which is free to use. Click *Review+create* to create the web app.



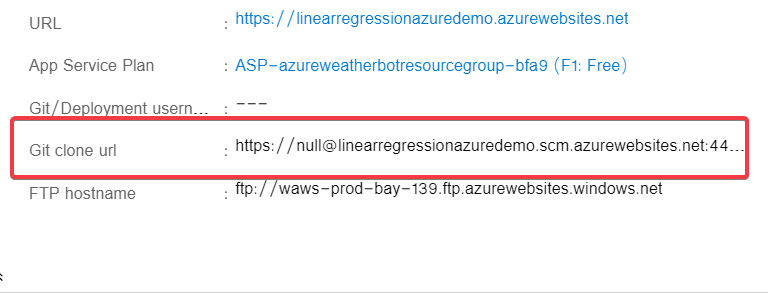
* Once the deployment is completed, open the app and go to the ‘Deployment Center’ option. Select ‘local git’ for source control and click continue.



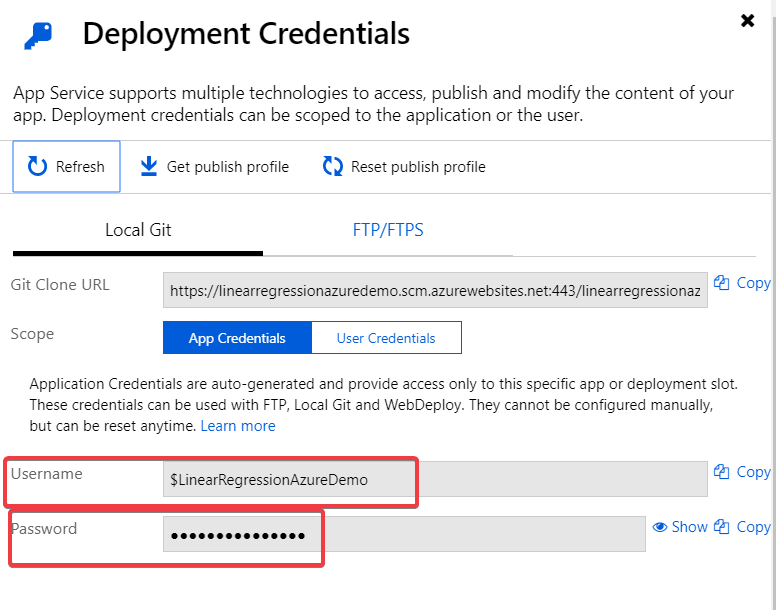
* Select the kudo ‘App service build provider’ as the build provider and click continue.



* Click ‘Finish’ to complete the setup.
* Go to the overview section of the app, and the Git link now will be visible.



* Go to ‘Deployment Credentials’ and copy the username and password. These will be required when doing the final push to the remote git repository.



* Open a command prompt and navigate to your project folder.
* Run git init to initialise an empty git repository
* Create a new remote git alias using the command: git remote add <alias> <git clone url>
* Use git add . to add all the files to the local git repository.
* Use git commit -m “First Commit” to commit the code to the git repo.
* Push the code to the remote repo using git push <alias> master –f
* This prompts for a username and password. Provide the same credentials as copied in the step above.

After deployment, from the ‘overview’ ­­­section, copy the URL and paste into the browser to see the application running.