

The Lightweight IBM Cloud Garage Method for Data Science

Architectural Decisions Document Template

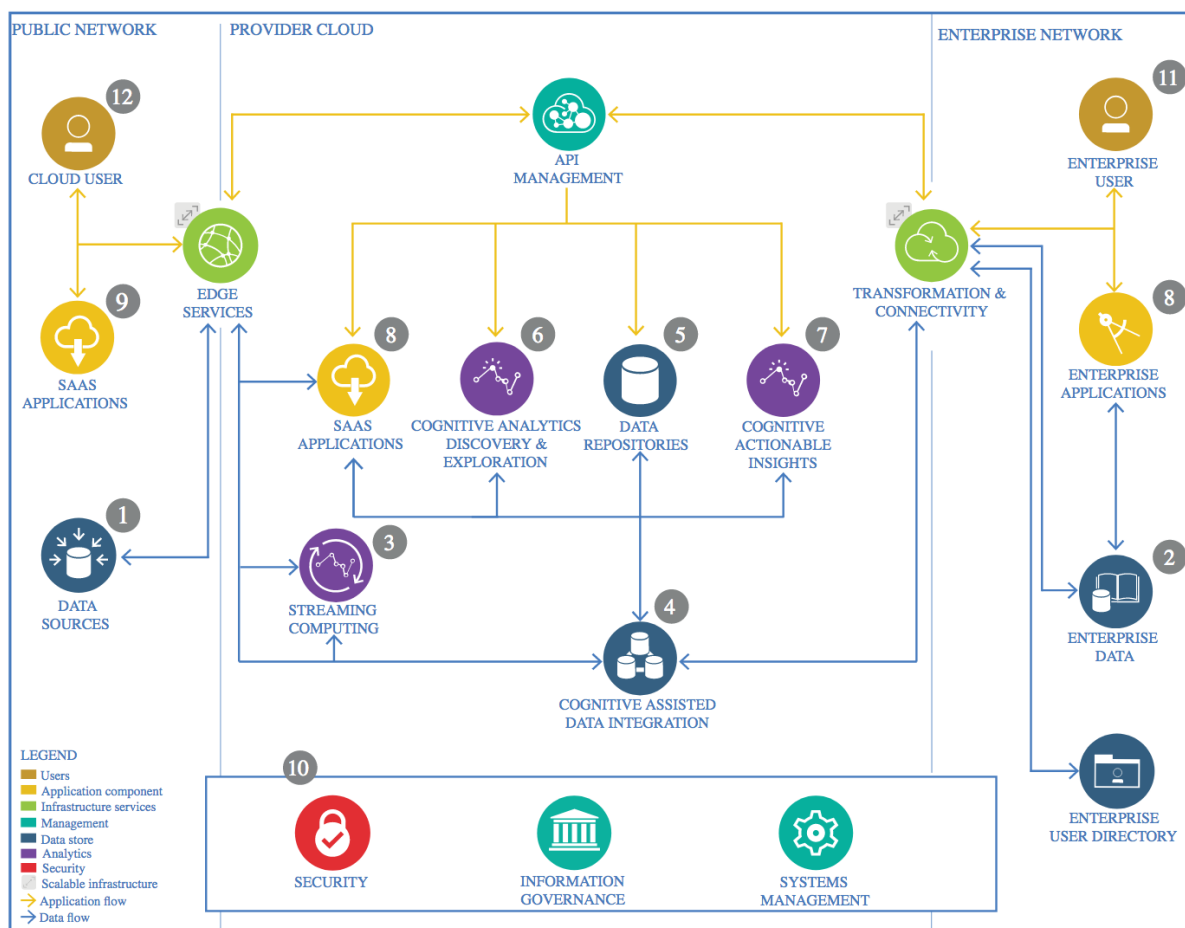
For this Capstone project I decided to solve image classification task by using Watson Machine Learning.

The task is to classify images with grape leaves diseases using different types of models. The given dataset contains 272 images which are labeled into 4 classes – dry, esca, healthy and mite classes.

I used Watson Studio environment for model creation, training and testing the models. Watson machine learning repository was used for deployment of these models.

The model_deployment_endpoint_url of the deployed model then can be used as REST API for scoring new images.

1 Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

1.1 Data Source

1.1.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.1.2 Justification

For this project I used dataset from the following link:

https://github.com/RenataUjhaziova/datasetWS_DvsEvshvsM_512-512_git

This dataset contains images with healthy grape leaves and with three different kind of diseases:

- “Dry grape leaves diseases”,
- Esca,
- Grape erineum mite disease.

The images were captured in one Vineyard in the South East of Slovakia and the dataset was prepared from these images.

The dataset contains images for training set, test set and validation set. The number of images is equal for each class that's why there was no need to solve issue with imbalanced dataset.

1.2 Enterprise Data

1.2.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.2.2 Justification

For this project data are loaded from Github repository or from Cloud object storage of the project in Watson studio.

But in the real-life use cases images will be captured by drones and stored in the Cloudant database. Then they will be loaded from this database and will send to the REST API of the deployed model on IBM Watson studio for classification.

1.3 Streaming analytics

1.3.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.3.2 Justification

Streaming analytics is not used in this project.

1.4 Data Integration

1.4.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.4.2 Justification

Data were integrated into IBM Watson studio in 2 ways:

- from Github repository and
- from IBM Cloud object storage of the project in Watson studio.

In IBM Watson studio I created Notebooks with support of Python 3.5 and I used python for data preprocessing and modeling.

Because I used different approaches for data preprocessing and data modeling, I used the following python libraries:

1. NumPy
2. Pandas
3. PIL
4. Scikit learn
5. Matplotlib
6. Seaborn
7. Xgboost
8. Keras
9. Watson-machine-learning-client

1.5 Data Repository

1.5.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.




1.5.2 Justification

In this project I used 2 ways for loading the dataset.

For training the xgboost model data was loaded from Github repository.

For training the CNN models data was loaded from Cloud object storage stored in IBM Watson studio.





Dataset contains 272 images in jpg format and image size 552 x 552 px with the following structure:

 test_set
 training_set
 validation_set

The total number of images for

- the training set is 200 images,
- the test set is 48 images,
- the validation set is 12 images.

In each dataset there are image collected in 4 different classes:

 dry
 esca
 healthy
 mite

The total number of images is the same for each class in the specific dataset.

1.6 Discovery and Exploration

1.6.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.6.2 Justification

The dataset was well structured and balanced that's why it was possible to skip the Data audit step.

For data understanding I used Color Histograms which showed the intensities of pixel color across different classes. For data exploration and data visualization I used libraries such as Cv2, numpy and matplotlib.

For data preparation phase pandas and scikit-learn libraries was used.

1.7 Actionable Insights

1.7.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.7.2 Justification

The given dataset contains just 272 images, which is not too much for training image classification models, that's why I decided to use different approaches.

The first approach was to use machine learning algorithms while the second approach was to use deep learning algorithms.

From machine learning algorithms the following models were used:

1. KNN

A) The accuracy of KNN model using **training and test set split** was: **80.65%**

```
# splitting the dataset into the Training and Testing set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
# Create the KNN classifier
KNN_model = KNeighborsClassifier()
```

B) Using **grid search** with the following parameters the accuracy of the best model was: **83.87%**

```
# Create KNN pipeline, set up parameter grid.
KNN_model_gs = KNeighborsClassifier()
parameters = {'n_neighbors': [4, 6, 8, 10],
              'weights': ['uniform', 'distance'],
              'metric': ['euclidean', 'manhattan']}
```

```
# Search for the best parameters.
clf = GridSearchCV(estimator = KNN_model_gs,
                   param_grid = parameters,
                   verbose = 1,
                   cv = 5,
                   n_jobs = -1)
```

2. XGboost model

A) The XGboost model trained with number of trees = 85 had accuracy: **85.48%**

```
# Select trained model.
n_trees = 85
y_pred = xgb_model.predict(X_test, ntree_limit= n_trees)
```

B) Using **PCA** for model tuning the accuracy of the model was: **88.71%**

```
pca = PCA(n_components=10)
xgb_model_pca = XGBClassifier(n_estimators=n_trees)
pipeline = Pipeline(steps=[('pca', pca), ('xgb', xgb_model_pca)])
```

From deep learning algorithms the following models were used:

1. ANN model

A) The accuracy of ANN model using **training and test set split** was: **83.87%**

```

ANN_model = Sequential()
# Adding the input layer and the first hidden layer
ANN_model.add(Dense(output_dim = 45, init = 'uniform', activation = 'relu', input_dim = 90))
# Adding the second hidden layer
ANN_model.add(Dense(output_dim = 45, init = 'uniform', activation = 'relu'))
# Adding the output layer
ANN_model.add(Dense(output_dim = 4, init = 'uniform', activation = 'softmax'))
# Compiling the ANN
ANN_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])

```

B) The accuracy of ANN model using **cross validation** was: **83.87%**

```

seed = 7
np.random.seed(seed)
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)

for train, test in kfold.split(X, y):
    cv_model = Sequential()
    cv_model.add(Dense(output_dim = 45, init = 'uniform', activation = 'relu', input_dim = 90))
    cv_model.add(Dense(output_dim = 45, init = 'uniform', activation = 'relu'))
    cv_model.add(Dense(output_dim = 4, init = 'uniform', activation = 'softmax'))

    # Compiling the ANN model
    cv_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])

    # Fitting the ANN model
    cv_model.fit(X_train, y_train, batch_size = 10, nb_epoch = 30, validation_data=(X_test, y_test),
        verbose=1)

```

2. CNN model using pre-trained VGG16 model – image size 224x224

The accuracy of the CNN model using **pre-trained VGG16 model** was:

- A) on the test set: **79.16%**
- B) on the validation set: **79.16%**

```

vgg16_model = keras.applications.vgg16.VGG16()

# Iterate over the functional layers and add it as a stack
model = Sequential()
for layer in vgg16_model.layers[:-1]:
    model.add(layer)

# Since the model is already trained with certain weights, we don't want to change it. Let it be the same
for layer in model.layers:
    layer.trainable = False

# Add the last layer
model.add(Dense(4, activation='softmax'))

# Compile the model

```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit_generator(train_batches, steps_per_epoch=20, validation_data=test_batches,  
validation_steps = 48, epochs=5, verbose=1, shuffle=False, callbacks=[plot_learning])
```

3. CNN sequential model – image size 128x128

The accuracy of the CNN sequential model:

- A) on the test set: **77.08%**
- B) on the validation set: **91.66%**

```
# Initialising the CNN
```

```
model = Sequential()
```

```
# Step 1 - Convolution
```

```
model.add(Conv2D(filters = 32, kernel_size=(3,3), input_shape = input_shape, activation = 'relu'))
```

```
# Step 2 - Pooling
```

```
model.add(MaxPooling2D(pool_size = (2, 2)))
```

```
# Adding a second convolution layer
```

```
model.add(Conv2D(filters = 64, kernel_size=(3,3), activation = 'relu'))
```

```
model.add(MaxPooling2D(pool_size = (2, 2)))
```

```
model.add(Dropout(rate=0.20))
```

```
# Adding a third convolution layer
```

```
model.add(Conv2D(filters = 128, kernel_size=(3,3), activation = 'relu'))
```

```
model.add(MaxPooling2D(pool_size = (2, 2)))
```

```
model.add(Dropout(rate=0.20))
```

```
# Step 3 - Flattening
```

```
model.add(Flatten())
```

```
# Step 4 - Full connection
```

```
model.add(Dense(units = 128, activation = 'relu'))
```

```
model.add(BatchNormalization())
```

```
model.add(Dropout(rate=0.5))
```

```
# Step 5 - Final layer
```

```
model.add(Dense(units = num_classes, activation = 'softmax'))
```

```
# Compile the model
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit_generator(train_batches, steps_per_epoch=7, validation_data=test_batches,  
validation_steps = 48, epochs=200, verbose=1, shuffle=False, callbacks=[plot_learning])
```

1.8 Applications / Data Products

1.8.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.8.2 Justification

The deployment of each model was done in IBM Watson studio using the Watson machine learning repository. Deployed models can be used for scoring new data by requesting the REST API = the scoring end point of each model.

This REST API can be used for image classification of the cloud - based application. Images will be captured by drones, stored in the Cloudant database, send to the REST API for classification and the location of images which were classified as dry, esca and mite will be displayed on the web application on the world map.

1.9 Security, Information Governance and Systems Management

1.9.1 Technology Choice

Please describe what technology you have defined here. Please justify below, why. In case this component is not needed justify below.

1.9.2 Justification

Security, Information Governance and System management was not a part of this project.