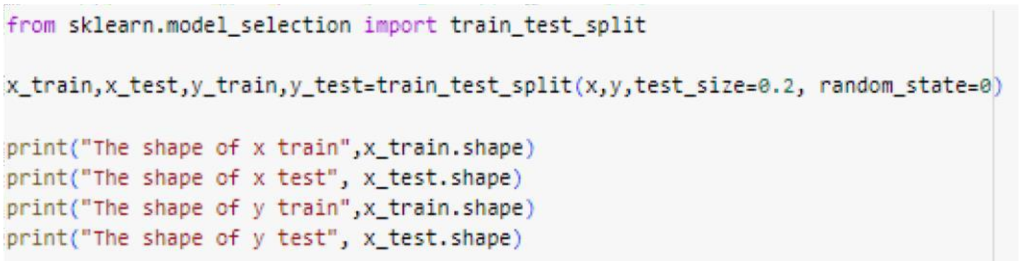
# Model Development Phase Template

| | |
|---|---|
| Date | 21 June 2024 |
| Team ID | 739792 |
| Project Title | Opticrop: Smart Agricultural Production Optimization Engine |
| Maximum Marks | 4 Marks |

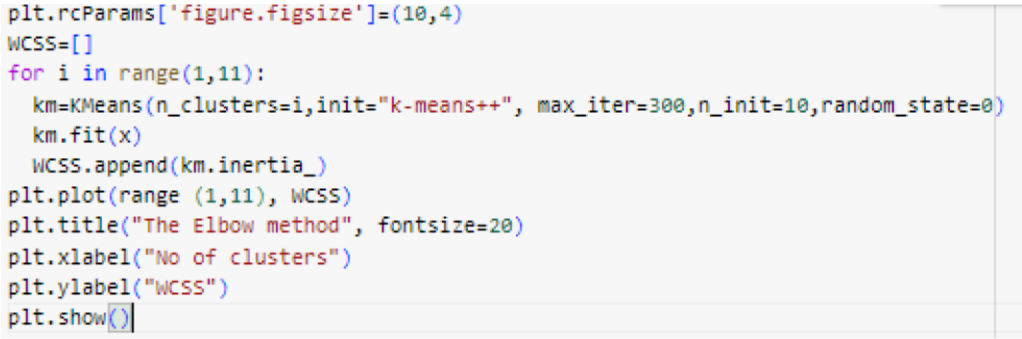
## Initial Model Training Code, Model Validation and Evaluation Report:

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

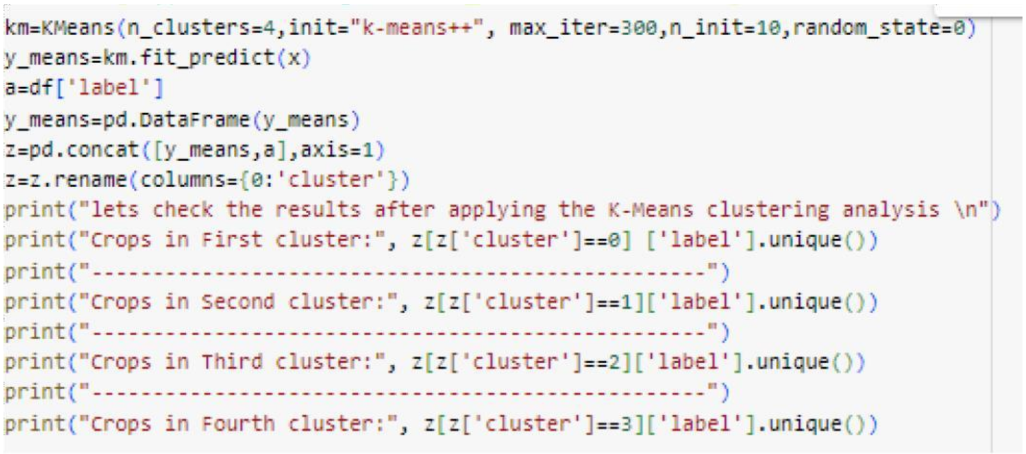
## Initial Model Training Code:



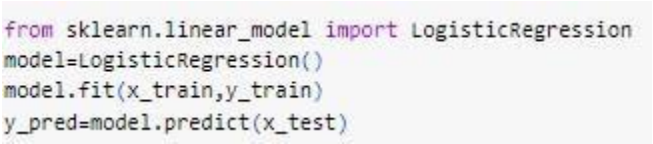
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=0)

print("The shape of x train",x_train.shape)
print("The shape of x test", x_test.shape)
print("The shape of y train",x_train.shape)
print("The shape of y test", x_test.shape)
```



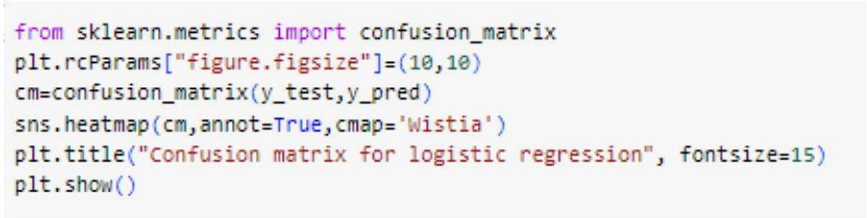
```
plt.rcParams['figure.figsize']=(10,4)
WCSS=[]
for i in range(1,11):
  km=KMeans(n_clusters=i,init="k-means++", max_iter=300,n_init=10,random_state=0)
  km.fit(x)
  WCSS.append(km.inertia_)
plt.plot(range (1,11), WCSS)
plt.title("The Elbow method", fontsize=20)
plt.xlabel("No of clusters")
plt.ylabel("WCSS")
plt.show()
```



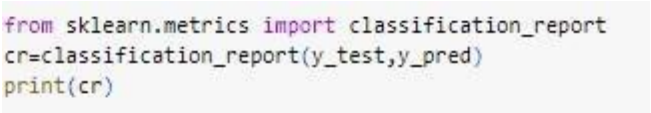
```
km=KMeans(n_clusters=4,init="k-means++", max_iter=300,n_init=10,random_state=0)
y_means=km.fit_predict(x)
a=df['label']
y_means=pd.DataFrame(y_means)
z=pd.concat([y_means,a],axis=1)
z=z.rename(columns={0:'cluster'})
print("lets check the results after applying the K-Means clustering analysis \n")
print("Crops in First cluster:", z[z['cluster']==0] ['label'].unique())
print("---------------------------------------------------")
print("Crops in Second cluster:", z[z['cluster']==1]['label'].unique())
print("---------------------------------------------------")
print("Crops in Third cluster:", z[z['cluster']==2]['label'].unique())
print("---------------------------------------------------")
print("Crops in Fourth cluster:", z[z['cluster']==3]['label'].unique())
```



```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```



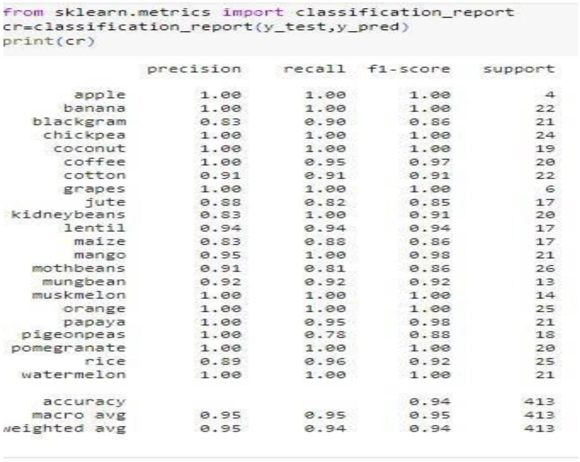
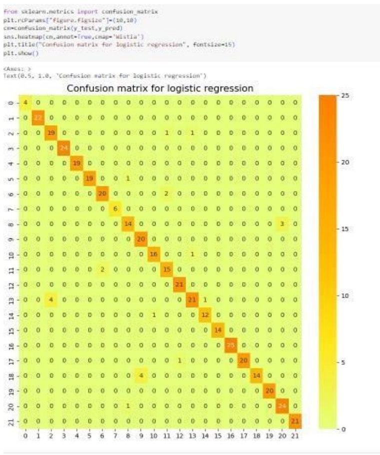
```
from sklearn.metrics import confusion_matrix
plt.rcParams["figure.figsize"]=(10,10)
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,cmap='Wistia')
plt.title("Confusion matrix for logistic regression", fontsize=15)
plt.show()
```



```
from sklearn.metrics import classification_report
cr=classification_report(y_test,y_pred)
print(cr)
```

### Model Validation and Evaluation Report:

| Model | Classification Report | F1 Score | Confusion Matrix |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Kmeans ,<br>Linear<br>regression |  | 94% |  |

```
from sklearn.metrics import classification_report
cr=classification_report(y_test,y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00         4
      banana       1.00      1.00      1.00        22
   blackgram       0.83      0.90      0.86        21
    chickpea       1.00      1.00      1.00        24
     coconut       1.00      1.00      1.00        19
      coffee       1.00      0.95      0.97        20
      cotton       0.91      0.91      0.91        22
      grapes       1.00      1.00      1.00         6
        jute       0.88      0.82      0.85        17
 kidneybeans       0.83      1.00      0.91        20
      lentil       0.94      0.94      0.94        17
       maize       0.83      0.88      0.86        17
       mango       0.95      1.00      0.98        21
   mothbeans       0.91      0.81      0.86        26
    mungbean       0.92      0.92      0.92        13
   muskmelon       1.00      1.00      1.00        14
      orange       1.00      1.00      1.00        25
      papaya       1.00      0.95      0.98        21
  pigeonpeas       1.00      0.78      0.88        18
 pomegranate       1.00      1.00      1.00        20
        rice       0.89      0.96      0.92        25
  watermelon       1.00      1.00      1.00        21

    accuracy                           0.94       413
   macro avg       0.95      0.95      0.95       413
weighted avg       0.95      0.94      0.94       413
```

```
from sklearn.metrics import confusion_matrix
plt.rcParams["figure.figsize"]=(10,10)
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,cmap='Wistia')
plt.title("Confusion matrix for logistic regression", fontsize=15)
plt.show()
```

```
<Axes: >
Text(0.5, 1.0, 'Confusion matrix for logistic regression')
```

Confusion matrix for logistic regression

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 19 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 21 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 21 |