```
# coding: utf-8

# # CASE STUDY: BREAST CANCER CLASSIFICATION
#

# # STEP #1: PROBLEM STATEMENT

#
# - Predicting if the cancer diagnosis is benign or malignant based on several
observations/features
# - 30 features are used, examples:
#         - radius (mean of distances from center to points on the perimeter)
#         - texture (standard deviation of gray-scale values)
#         - perimeter
#         - area
#         - smoothness (local variation in radius lengths)
#         - compactness (perimeter^2 / area - 1.0)
#         - concavity (severity of concave portions of the contour)
#         - concave points (number of concave portions of the contour)
#         - symmetry
#         - fractal dimension ("coastline approximation" - 1)
#
# - Datasets are linearly separable using all 30 input features
# - Number of Instances: 569
# - Class Distribution: 212 Malignant, 357 Benign
# - Target class:
#          - Malignant
#          - Benign
#
#
# https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
#
# ![image.png](attachment:image.png)

# # STEP #2: IMPORTING DATA

# In[1]:


# import libraries
import pandas as pd # Import Pandas for data manipulation using dataframes
import numpy as np # Import Numpy for data statistical analysis
import matplotlib.pyplot as plt # Import matplotlib for data visualisation
import seaborn as sns # Statistical data visualization
# %matplotlib inline


# In[2]:


# Import Cancer data drom the Sklearn library
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()


# In[3]:
```

```
cancer


# In[4]:


cancer.keys()


# In[5]:


print(cancer['DESCR'])


# In[6]:


print(cancer['target_names'])


# In[7]:


print(cancer['target'])


# In[8]:


print(cancer['feature_names'])


# In[9]:


print(cancer['data'])


# In[10]:


cancer['data'].shape


# In[11]:


df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns =
np.append(cancer['feature_names'], ['target']))


# In[12]:


df_cancer.head()


# In[13]:
```

```
df_cancer.tail()


# In[14]:


x = np.array([1,2,3])
x.shape


# In[15]:


Example = np.c_[np.array([1,2,3]), np.array([4,5,6])]
Example.shape


# # STEP #3: VISUALIZING THE DATA

# In[16]:


sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture',
'mean area', 'mean perimeter', 'mean smoothness'] )


# In[17]:


sns.countplot(df_cancer['target'], label = "Count")


# In[18]:


sns.scatterplot(x = 'mean area', y = 'mean smoothness', hue = 'target', data =
df_cancer)


# In[19]:


#sns.lmplot('mean area', 'mean smoothness', hue ='target', data = df_cancer_all,
fit_reg=False)


# In[20]:


# Let's check the correlation between the variables
# Strong correlation between the mean radius and mean perimeter, mean area and mean
primeter
plt.figure(figsize=(20,10))
sns.heatmap(df_cancer.corr(), annot=True)


# # STEP #4: MODEL TRAINING (FINDING A PROBLEM SOLUTION)
```

```
# In[21]:


# Let's drop the target label coloumns
X = df_cancer.drop(['target'],axis=1)


# In[22]:


X


# In[23]:


y = df_cancer['target']
y


# In[24]:


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state=5)


# In[25]:


X_train.shape


# In[26]:


X_test.shape


# In[27]:


y_train.shape


# In[28]:


y_test.shape


# In[29]:


from sklearn.svm import SVC
```

```
from sklearn.metrics import classification_report, confusion_matrix

svc_model = SVC()
svc_model.fit(X_train, y_train)


# # STEP #5: EVALUATING THE MODEL

# In[30]:


y_predict = svc_model.predict(X_test)
cm = confusion_matrix(y_test, y_predict)


# In[31]:


sns.heatmap(cm, annot=True)


# In[32]:


print(classification_report(y_test, y_predict))


# # STEP #6: IMPROVING THE MODEL

# In[33]:


min_train = X_train.min()
min_train


# In[34]:


range_train = (X_train - min_train).max()
range_train


# In[35]:


X_train_scaled = (X_train - min_train)/range_train


# In[36]:


X_train_scaled


# In[37]:


sns.scatterplot(x = X_train['mean area'], y = X_train['mean smoothness'], hue =
```

```
y_train)


# In[38]:


sns.scatterplot(x = X_train_scaled['mean area'], y = X_train_scaled['mean smoothness'], hue = y_train)


# In[39]:


min_test = X_test.min()
range_test = (X_test - min_test).max()
X_test_scaled = (X_test - min_test)/range_test


# In[40]:


from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

svc_model = SVC()
svc_model.fit(X_train_scaled, y_train)


# In[41]:


y_predict = svc_model.predict(X_test_scaled)
cm = confusion_matrix(y_test, y_predict)

sns.heatmap(cm,annot=True,fmt="d")


# In[42]:


print(classification_report(y_test,y_predict))


# # IMPROVING THE MODEL - PART 2

# In[43]:


param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel': ['rbf']}


# In[44]:


from sklearn.model_selection import GridSearchCV


# In[45]:
```

```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=4)

# In[46]:

grid.fit(X_train_scaled,y_train)

# In[47]:

grid.best_params_

# In[48]:

grid.best_estimator_

# In[49]:

grid_predictions = grid.predict(X_test_scaled)

# In[50]:

cm = confusion_matrix(y_test, grid_predictions)

# In[51]:

sns.heatmap(cm, annot=True)

# In[52]:

print(classification_report(y_test,grid_predictions))

# # Excellent Job!
```