In [ ]:
```
# Importing the Libraries
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error, log_loss
from tqdm import tqdm_notebook
import seaborn as sns
import imageio
import time
from IPython.display import HTML

from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.datasets import make_blobs
```

In [ ]:
```
# Importing the Data
data = pd.read_csv('Final_Refined_Encoded.csv')
```

In [ ]:
```
data.head(20)
```

In [ ]:
```
# Changing the Labels for categories
bins = [1, 2, 3, 4]
#bins = [0, 50, 150, 250, 500, 1000]
names = [0, 1, 2, 3]

label_dict = dict(enumerate(names, 1))
price = pd.Series(np.vectorize(label_dict.get)(np.digitize(data['Price'], bir
price
```

In [ ]:
```
data['Price'] = price
```

In [ ]:
```
# Saperating the Target Column
X, y = data.iloc[:, :-1], data.iloc[:, -1]
```

In [ ]:
```
# Test-Train Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ranc
```

In [ ]:
```
# Standard-Scalar
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [ ]:
```
# OneHotEncoding the output categories
enc = OneHotEncoder()
# 1 -> (1, 0, 0, 0, 0, 0), 2 -> (0, 1, 0, 0, 0, 0), 3 -> (0, 0, 1, 0, 0, 0),
y_OH_train = enc.fit_transform(np.expand_dims(y_train,1)).toarray()
y_OH_val = enc.fit_transform(np.expand_dims(y_test,1)).toarray()
print(y_OH_train.shape, y_OH_val.shape)
```

In [ ]:
```
# Innitizing the Classifier
clf = MLPClassifier(activation='relu', solver='adam', alpha=0.00001, momentum
```

In [ ]:
```
# Training the NeuralNetwork
clf.fit(X_train, y_OH_train)
```

In [ ]:
```
# Accuracy Calculation and Printing
Y_pred_train = clf.predict(X_train)
Y_pred_train = np.argmax(Y_pred_train,1)

Y_pred_val = clf.predict(X_test)
Y_pred_val = np.argmax(Y_pred_val,1)

accuracy_train = accuracy_score(Y_pred_train, y_train)
accuracy_val = accuracy_score(Y_pred_val, y_test)

print("Training accuracy", round(accuracy_train, 2))
print("Validation accuracy", round(accuracy_val, 2))
```

In [ ]:
```

```