Subject: Applied Data Science (DJ19DSL703)

Experiment - 7

(Modelling and Optimisation Trade-off)

Name: Jhanvi Parekh   **SAP ID: 60009210033**   **Batch: D11**

Aim: To optimize the model.

Theory:

Hyper parameter and parameter tuning are crucial processes in machine learning that help optimize the performance of a model. They involve adjusting the settings of a machine learning algorithm to improve its ability to make accurate predictions. The 'K' in KNN can be regarded as the parameter whereas number of leaf nodes, number of branches, depth of a tree are hyper parameters of a Decision Tree or Random Forest.

1. Parameters:
Parameters are the internal settings or coefficients learned by the machine learning model during training. These values are adjusted automatically through the training process, and they determine how the model transforms input data into predictions. In a simple linear regression model, for example, the parameters are the slope and intercept of the regression line. Parameters are learned from the data and are specific to the model and its training data.

2. Hyper parameters:
Hyper parameters are settings or configurations that are not learned from the data but are set by the machine learning engineer or data scientist before training begins.
These settings control the behaviour of the learning algorithm itself and can significantly impact the model's performance.

Examples of hyper parameters include the learning rate in gradient descent, the number of hidden layers and neurons in a neural network, or the regularization strength in a support vector machine.
Hyper parameters are set by trial and error or by using techniques like grid search or random search to find the best values for a specific problem.

Parameter Tuning:
Parameter tuning involves adjusting the internal parameters of a machine learning model to optimize its performance on a specific dataset. This is typically done automatically during the training process, as the model learns the best values for its parameters to minimize a predefined loss function.
For example, in a neural network, the weights and biases of the neurons are adjusted during training to minimize the prediction error. Parameter tuning is specific to the model and is intrinsic to the training process.

Hyper parameter Tuning:

Hyper parameter tuning, also known as hyper parameter optimization, focuses on finding the best settings for hyper parameters to improve a model's generalization and performance. Hyper parameter tuning is an iterative process that typically involves searching through a range of hyper parameter values to find the combination that produces the best results on a validation dataset. Common techniques for hyper parameter tuning include grid search, random search, Bayesian optimization, and automated tools like scikit-learn's GridSearchCV or hyperopt.

Hyper parameter tuning is crucial for optimizing the performance of a machine learning model on a specific problem. In summary, parameters are internal settings learned by the model during training, while hyper parameters are external settings configured before training. Parameter tuning focuses on optimizing internal settings, while hyper parameter tuning is about finding the best external settings to improve model performance. Both processes are essential for developing effective machine learning models.

Lab Assignment:

- Students need to try various permutations and combinations for getting the right set of parameters and hyper parameter values.
- Post this your model is ready for deployment. Create a pickle file for your model such that it will take the input and produce the model response as the output. This should be then converted to an API using Flask/FastAPI frameworks.
- The API will have an endpoint (say /run-model) which will be a POST request. It will take the data records for which the prediction is to be made as input, run the model in the background and the API response will be the output from the model. For example, I can provide an email body as input to the API and it will return a response if the content can be regarded as spam or not. Or I can even provide multiple input fields for a house description and the endpoint will provide the estimated house price in response.

```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
import pandas as pd
import re
from sklearn.feature_extraction.text import CountVectorizer

                    Loading...

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))
# cv = pickle.load(open('cv.pkl','rb'))
data = pd.read_table(r'C:/Users/hjain/Downloads/SMSSpamCollection.txt', sep='\t', names=["label", "message"

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    message = request.form.get('msg') # Assuming the input field name in your form is 'rate'
    cv = CountVectorizer(max_features=2500, binary=True)
    cv.fit_transform(data['message'])
    X = cv.transform([message])  # Use the CountVectorizer to transform the input
    prediction = model.predict(X)
    if prediction[0] == True:
```

```python
    if prediction[0] == True:
        prediction_text = "The message is classified as spam"
    else:
        prediction_text = "The message is classified as ham (non-spam)"

    return render_template('index.html', prediction_text=prediction_text)

f __name__ == "__main__":
    app.run(debug=True)
```

```html
</head>

<body style="background: lightblue;">
 <div class="login">
  <h1>Sales Forecasting</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
     <input type="text" name="msg" placeholder="msg" required="required" />
      <!-- <input type="text" name="sales in first month" placeholder="sales in first month" required="required
    <input type="text" name="sales in second month" placeholder="sales in second month" required="required" /> --
      <button type="submit" class="btn btn-primary btn-block btn-large">Classify as spam or ham</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>
</body>
</html>
```

| We have a package for you | Classify as spam or ham |

The message is classified as spam

Conclusion: In this lab assignment, we experimented with various parameter and hyperparameter configurations to optimize model performance. After achieving the desired results, the model was serialized using a pickle file to enable seamless deployment. Finally, the model was integrated into an API using Flask to allow for real-time interaction and prediction.