Experiment No. : 5 Page No. : 11

HYPERPARAMETER TUNING

Aim: To build a machine learning model to predict the likelyhood of diabetes in patients based on various health parameters. The model is to be developed using hyperparameter tuning to improve prediction accuracy.

Libraries:

1) Pandas: For data loading, manipulation and preprocessing
2) numpy: For numerical operations and handling arrays.
3) matplotlib.pyplot and seaborn: for data visualisation
4) Sklearn: multiple models:
   a) ensemble, linear_model, tree: Different Classifiers (Random forest, Logestic Regression, Decision Tree, and etc)
   b) metrics: Performance metrics.
   c) model_selection: Spliting and performing cross validation on data
   d) preprocessing: For scaling and transforming features.
5) scipy.stats: For statistical functions.

1) Reading the data:

```
df = pd.read_csv("diabetes.csv");
```

2) Cleaning Data:

```
df.dropna(inplace = True)
```

Teacher's Signature :

```
df.fillna(0, inplace = True)
df.drop_duplicates(inplace = True)
```

3) Training and Testing Split

```
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 = 0).
```

4) Linear Regression:

```
model = lm().fit(x_train, y_train)
pred = model.predict(x_test)
plt.scatter(y_test, pred)
plt.xlabel("True values").
plt.ylabel('prediction')
plt.show();
print('accuracy:', model.score(x_test, y_test))
```

5) Random forest

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(random_state=20, criterion='
        entropy', n_estimators=20)
rf.fit(x_train, y_train)
```

6) Hyperparameter Tuning Techniques:

a) Grid Search:

```
from sklearn.model_selection import GridSearchCV
```

Teacher's Signature : _____

```
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [None, 10, 20, 30]}
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 5, n_jobs = -1, verbose = 2)
grid_search.fit(x_train, y_train)
print("Best parameters found:", grid_search.best_params_)
```

b) Randomized Search:

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_dist = {
    'n_estimators': randint(10, 100),
    'max_depth': [None, 10, 20, 30]
}

random_search = RandomizedSearchCV(estimator = rf, param_distributions
                = param_dist, n_iter = 10, cv = 5, n_jobs = -1, verbose = 2)
random_search.fit(x_train, y_train)
print("Best parameters found:", random_search.best_params_)
```

c) Bayesian Optimization

```
from skopt import BayesSearchCV
from skopt.space import Integer
param_grid = {
    'n_estimators': Integer(10, 100),
    'max_depth': Integer(10, 30)
}
```

Teacher's Signature : _____

bayes_search = BayesSearchCV(estimator=rf, search_spaces=param_grid,
n_iter=32, cv=5, n_jobs=-1, verbose=2)
bayes_search.fit(x_train, y_train)
print("Best Performance found:", bayes_search.best_params_)

~~d) Hyp~~

7) Result Visualization:

import seaborn as sns
import matplotlib.pyplot as plt
fig, axes = plt.subplots(1, 2, fig size=(10, 5), sharey=True)
sns.boxplot(x="Outcome", y="Blood pressure", data=df,
ax=axes[0])
sns.violinplot(x="Outcome", y="BloodPressure", data=df,
ax=axes[1])