```
# 1. load iris dataset
# 2. identify the distinct hyperparameters for logistic regression model. Use "one vs rest"
# 3. Tune hyperparameter values and find best parameters
# 4. plot accuracy verses distinct values of hyperparameters of logistic regression model versions
# 5. For the best model predict test performance. Plot confusion matrix
# 6. Repeat the same for Indian pima dataset
```

```
#On Iris Data-set
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
# Load Iris Dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

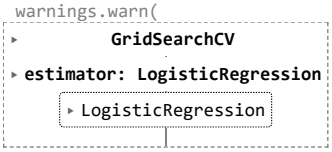
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Define the model
model = LogisticRegression(multi_class='ovr')
```

```
# Define the hyperparameters to tune
param_grid = {'C': [0.1, 1, 10], 'penalty': ['l1', 'l2']}
```

```
# Grid Search Cross Validation
grid = GridSearchCV(model, param_grid, cv=5)
grid.fit(X_train, y_train)
```

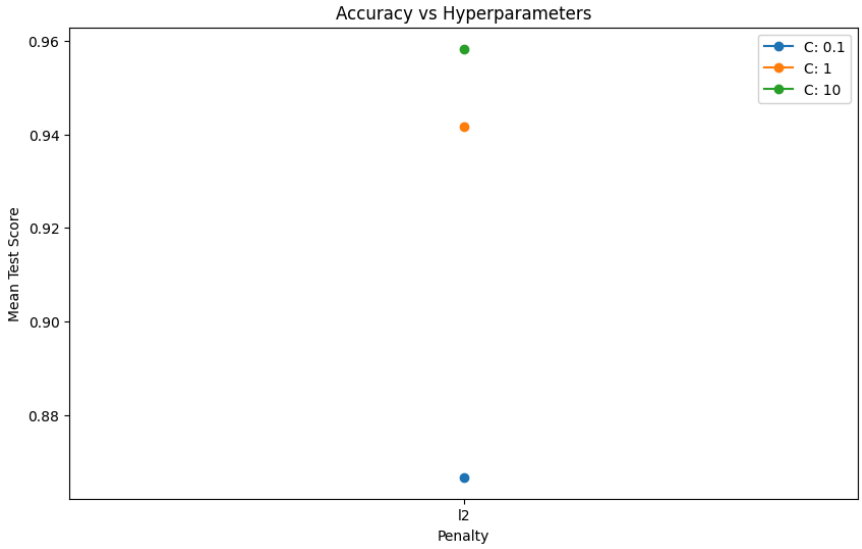
```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:378: F
15 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score=

Below are more details about the failures:
--------------------------------------------------------------------------------
15 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.p
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py", l
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py", l
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserW
  warnings.warn(
```




```
# Best parameters
print("Best Parameters: ", grid.best_params_)
```

```
Best Parameters:  {'C': 10, 'penalty': 'l2'}
```

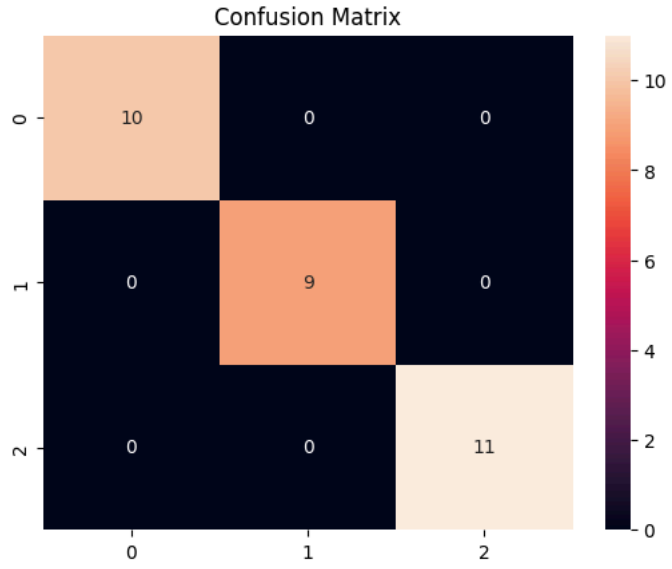
```
# Accuracy for each hyperparameter combination
results = pd.DataFrame(grid.cv_results_)
plt.figure(figsize=(10, 6))
for i, param in enumerate(param_grid['C']):
    temp = results[results['param_C'] == param]
    plt.plot(temp['param_penalty'], temp['mean_test_score'], marker='o', label='C: ' + str(param))
plt.legend()
plt.xlabel('Penalty')
plt.ylabel('Mean Test Score')
plt.title('Accuracy vs Hyperparameters')
plt.show()
```




```
# Predict and evaluate the best model
y_pred = grid.predict(X_test)
print("Test Accuracy: ", accuracy_score(y_test, y_pred))
```

```
Test Accuracy:  1.0
```

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.show()
```

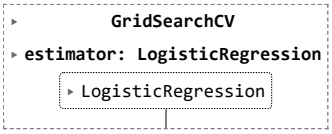


```
#ON Indian Pima diabetes Dataset
# 1. Load Indian Pima dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pd.read_csv(url, names=names)
```

```
# Prepare data
X = data.iloc[:,0:8]
y = data.iloc[:,8]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)
```

```
# 2. Identify distinct hyperparameters for Logistic Regression Model
# For Logistic Regression, some of the hyperparameters are 'C', 'penalty', 'fit_intercept'
parameters = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2'], 'fit_intercept': [True, False]}
```

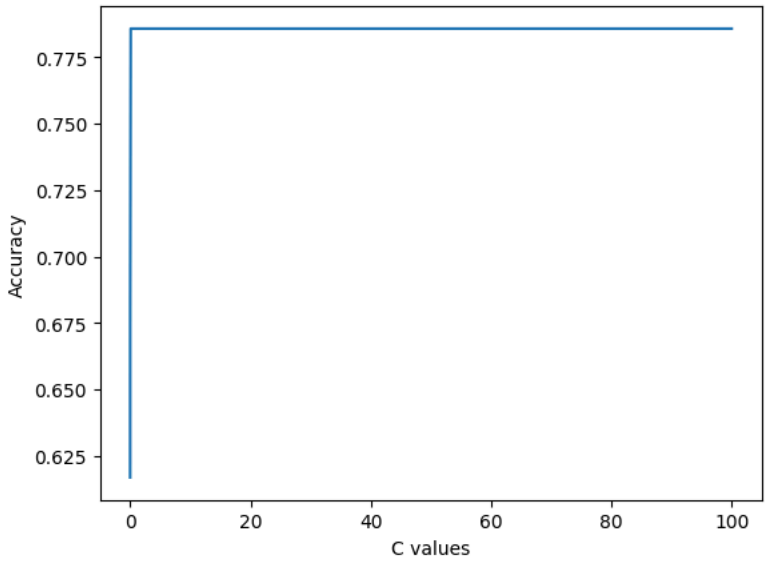
```
# 3. Tune hyperparameter values and find best parameters
logistic = LogisticRegression(solver='liblinear')
clf = GridSearchCV(logistic, parameters, cv=5)
clf.fit(X_train, y_train)
```




```
# Print best parameters
print("Best Parameters: ", clf.best_params_)
```

```
Best Parameters:  {'C': 10, 'fit_intercept': True, 'penalty': 'l1'}
```

```
# 4. Plot accuracy Vs distinct values of hyperparameters of logistic regression model versions
# Here, we plot for 'C' parameter
C_values = [0.001, 0.01, 0.1, 1, 10, 100]
accuracy_scores = []

for c in C_values:
    model = LogisticRegression(C=c, penalty=clf.best_params_['penalty'], fit_intercept=clf.best_params_['fit_intercept'], solver='libli
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

plt.plot(C_values, accuracy_scores)
plt.xlabel('C values')
plt.ylabel('Accuracy')
plt.show()
```




```
# 5. For the best model, predict best performance, find confusion matrix
best_model = LogisticRegression(C=clf.best_params_['C'], penalty=clf.best_params_['penalty'], fit_intercept=clf.best_params_['fit_inter
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)
```

```
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
```

```
Accuracy:  0.7857142857142857
Confusion Matrix:
 [[89  8]
 [25 32]]
```

```
#conclusion:
# The logistic regression model's performance varied significantly between the two datasets.
# With the Iris dataset, the model achieved perfect accuracy (1.0), indicating that the model could
# flawlessly classify the different Iris species. This could be due to the Iris dataset's relatively simple
# and well-separated classes. However, the model's accuracy dropped to 0.78 with the Indian Pima Diabetes dataset.
# This lower accuracy could be attributed to the complexity of the dataset, which may contain overlapping or less distinguishable classes
# or it could be due to the presence of more noise and outliers in the data.
# These results underscore the importance of understanding the characteristics of the dataset when evaluating a model's performance.
```