The programming assignment for the Logistic Regression model is

Become familiar with the Scikit-Learn toolkit. All of the documentation for this great machine learning toolkit is available online.

Use the Iris dataset that you used in the programming assignment for the Perceptron/Adaline module.

Using the Scikit-Learn Library, train the Logistic Regression model using the following:

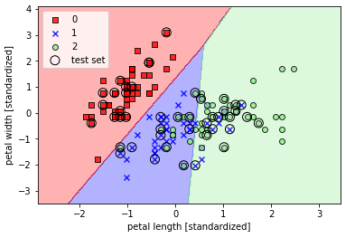
```
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):
    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
    np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=colors[idx],
                    marker=markers[idx], label=cl,
                    edgecolor='black')
    # highlight test samples
    if test_idx:
    # plot all samples
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1],
                    c='', edgecolor='black', alpha=1.0,
                    linewidth=1, marker='o',
                    s=100, label='test set')


#training perceptron
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
accuracy_table = []
l1_accuracy = []
l2_accuracy = []
```

All six cases of using two features at a time

Sepal length / Sepal width

```
#@title Sepal length / Sepal width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined_std, y_combined, classifier=lr, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy

res = ["Sepal length / Sepal width", accuracy]
accuracy_table.append(res)
```

```
<ipython-input-1-c83cc951b7c7>:27: MatplotlibDeprecationWarning: Using a string of single character colors as a color sequence is deprecated. Use an explicit list instead.
  plt.scatter(X_test[:, 0], X_test[:, 1],
```

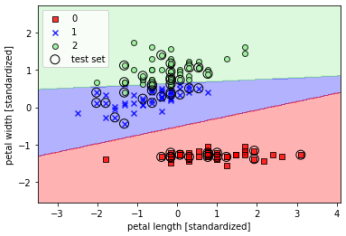


Sepal width / Petal Length

```
#@title Sepal width / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
```

```
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined_std, y_combined, classifier=lr, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal width / Petal Length", accuracy]
accuracy_table.append(res)
```

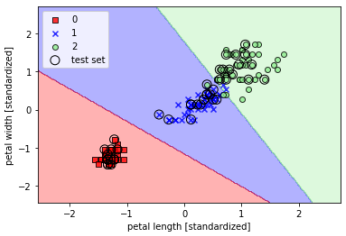
```
<ipython-input-1-c83cc951b7c7>:27: MatplotlibDeprecationWarning: Using a string of single character colors as a color sequence is deprecated. Use an explicit list instead.
  plt.scatter(X_test[:, 0], X_test[:, 1],
```



Petal Length / Petal Width

```
#@title Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined_std, y_combined, classifier=lr, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Petal Length / Petal Width", accuracy]
accuracy_table.append(res)
```

```
<ipython-input-1-c83cc951b7c7>:27: MatplotlibDeprecationWarning: Using a string of single character colors as a color sequence is deprecated. Use an explicit list instead.
  plt.scatter(X_test[:, 0], X_test[:, 1],
```

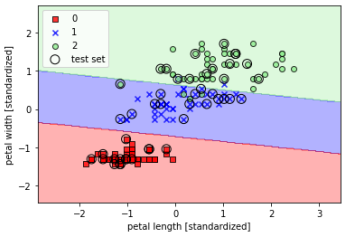


Sepal Length / Petal Width

```
#@title Sepal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined_std, y_combined, classifier=lr, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()

y_hat = lr.predict(X_combined_std)
```

```
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Width", accuracy]
accuracy_table.append(res)
```

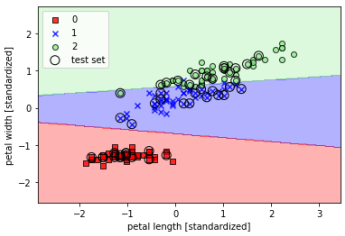
```
<ipython-input-1-c83cc951b7c7>:27: MatplotlibDeprecationWarning: Using a string of single character colors as a color sequence is deprecated. Use an explicit list instead.
  plt.scatter(X_test[:, 0], X_test[:, 1],
```



Sepal Length / Petal Length

```
#@title Sepal Length / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined_std, y_combined, classifier=lr, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Length", accuracy]
accuracy_table.append(res)
```

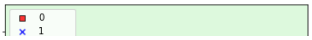
```
<ipython-input-1-c83cc951b7c7>:27: MatplotlibDeprecationWarning: Using a string of single character colors as a color sequence is deprecated. Use an explicit list instead.
  plt.scatter(X_test[:, 0], X_test[:, 1],
```



Sepal Width / Petal Width

```
#@title Sepal Width / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined_std, y_combined, classifier=lr, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Width / Petal Width", accuracy]
accuracy_table.append(res)
```

```
<ipython-input-1-c83cc951b7c7>:27: MatplotlibDeprecationWarning: Using a string of single character colors as a color sequence is deprecated. Use an explicit list instead.
  plt.scatter(X_test[:, 0], X_test[:, 1],
```



All four cases of using three features at a time

Sepal length / Sepal width / Petal Length

```
#@title Sepal length / Sepal width / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0,1,2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Length", accuracy]
accuracy_table.append(res)
```

Sepal length / Sepal width / Petal Width

```
#@title Sepal length / Sepal width / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Width", accuracy]
accuracy_table.append(res)
```

Sepal Length / Petal Length / Petal Width

```
#@title Sepal Length / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Length / Petal Width", accuracy]
accuracy_table.append(res)
```

Sepal Width / Petal Length / Petal Width

```
#@title Sepal Width / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Width / Petal Length / Petal Width", accuracy]
accuracy_table.append(res)
```

The one case of using all features at once

Sepal length / Sepal width / Petal Length / Petal Width

```
#@title Sepal length / Sepal width / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Length / Petal Width", accuracy]
accuracy_table.append(res)
```

Summarize your results (i.e, what's the best accuracy you can obtain for each of the 11 cases you considered, how many iterations does it take to converge, anything else you think is relevant and important) in a table

As a result, the best accuracy one can obtain from the 11 cases test above is the case with all four features. This is because BLANK. It took BLANK iterations to converge. Below is a table of results:

```
#table of results from 11 cases.
import pandas as pd
accuracy_df = pd.DataFrame(accuracy_table, columns=['Features', 'Accuracy'])
accuracy_df.style.set_caption("11 cases output")
```

11 cases output

	Features	Accuracy
0	Sepal length / Sepal width	0.813333
1	Sepal width / Petal Length	0.953333
2	Petal Length / Petal Width	0.960000
3	Sepal Length / Petal Width	0.960000
4	Sepal Length / Petal Length	0.960000
5	Sepal Width / Petal Width	0.953333
6	Sepal length / Sepal width / Petal Length	0.953333
7	Sepal length / Sepal width / Petal Width	0.960000
8	Sepal Length / Petal Length / Petal Width	0.960000
9	Sepal Width / Petal Length / Petal Width	0.980000
10	Sepal length / Sepal width / Petal Length / Petal Width	0.986667

Play with both L1 and L2 regularization and vary the regularization parameter C

Testing L1 regularization

```
#@title Testing L1 regularization
#Sepal length / Sepal width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
```

```
y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy

res = ["Sepal length / Sepal width", accuracy]
l1_accuracy


#Sepal width / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))


y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal width / Petal Length", accuracy]
l1_accuracy.append(res)


#Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Petal Length / Petal Width", accuracy]
l1_accuracy.append(res)


#Sepal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Width", accuracy]
l1_accuracy.append(res)


#Sepal Length / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
```

```
y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Length", accuracy]
l1_accuracy.append(res)

#Sepal Width / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Width / Petal Width", accuracy]
l1_accuracy.append(res)


#Sepal length / Sepal width / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0,1,2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Length", accuracy]
l1_accuracy.append(res)



#Sepal length / Sepal width / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Width", accuracy]
l1_accuracy.append(res)



#Sepal Length / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
```

```
y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Length / Petal Width", accuracy]
l1_accuracy.append(res)


#Sepal Width / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Width / Petal Length / Petal Width", accuracy]
l1_accuracy.append(res)


#Sepal length / Sepal width / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L1 penalty
lr = LogisticRegression(penalty='l1', C=1.0, random_state=1, solver='liblinear')
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Length / Petal Width", accuracy]
l1_accuracy.append(res)


#table of results from 11 cases.
import pandas as pd
l1_accuracy_df = pd.DataFrame(l1_accuracy, columns=['Features', 'Accuracy'])
l1_accuracy_df.style.set_caption("Adjusted L1")
```

Adjusted L1

	Features	Accuracy
0	Sepal width / Petal Length	0.906667
1	Petal Length / Petal Width	0.946667
2	Sepal Length / Petal Width	0.906667
3	Sepal Length / Petal Length	0.900000
4	Sepal Width / Petal Width	0.920000
5	Sepal length / Sepal width / Petal Length	0.900000
6	Sepal length / Sepal width / Petal Width	0.913333
7	Sepal Length / Petal Length / Petal Width	0.920000
8	Sepal Width / Petal Length / Petal Width	0.933333
9	Sepal length / Sepal width / Petal Length / Petal Width	0.933333

Testing L2 regularization

```
#@title Testing L2 regularization
l2_accuracy = []
#Sepal length / Sepal width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
```

```
y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy

res = ["Sepal length / Sepal width", accuracy]
l2_accuracy.append(res)


#Sepal width / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))


y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal width / Petal Length", accuracy]
l2_accuracy.append(res)


#Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Petal Length / Petal Width", accuracy]
l2_accuracy.append(res)


#Sepal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Width", accuracy]
l2_accuracy.append(res)


#Sepal Length / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
```

```
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Length", accuracy]
l2_accuracy.append(res)

#Sepal Width / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Width / Petal Width", accuracy]
l2_accuracy.append(res)


#Sepal length / Sepal width / Petal Length
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0,1,2]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Length", accuracy]
l2_accuracy.append(res)



#Sepal length / Sepal width / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Width", accuracy]
l2_accuracy.append(res)



#Sepal Length / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
```

```
y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Length / Petal Length / Petal Width", accuracy]
l2_accuracy.append(res)


#Sepal Width / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [1, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal Width / Petal Length / Petal Width", accuracy]
l2_accuracy.append(res)


#Sepal length / Sepal width / Petal Length / Petal Width
from sklearn import datasets
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data[:, [0, 1, 2, 3]]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)


from sklearn.linear_model import LogisticRegression
#adding L2 penalty
lr = LogisticRegression(penalty='l2', C=1.0, random_state=1)
lr.fit(X_train_std, y_train)
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))

y_hat = lr.predict(X_combined_std)
# Calculate accuracy on training data
accuracy=np.mean(y_hat==y_combined)
accuracy
res = ["Sepal length / Sepal width / Petal Length / Petal Width", accuracy]
l2_accuracy.append(res)


#table of results from 11 cases.
import pandas as pd
l2_accuracy_df = pd.DataFrame(l2_accuracy, columns=['Features', 'Accuracy'])
l2_accuracy_df.style.set_caption("Adjusted L2")
```

Adjusted L2

	Features	Accuracy
0	Sepal length / Sepal width	0.826667
1	Sepal width / Petal Length	0.940000
2	Petal Length / Petal Width	0.953333
3	Sepal Length / Petal Width	0.940000
4	Sepal Length / Petal Length	0.880000
5	Sepal Width / Petal Width	0.953333
6	Sepal length / Sepal width / Petal Length	0.913333
7	Sepal length / Sepal width / Petal Width	0.960000
8	Sepal Length / Petal Length / Petal Width	0.960000
9	Sepal Width / Petal Length / Petal Width	0.966667
10	Sepal length / Sepal width / Petal Length / Petal Width	0.973333

Discuss your findings. Does using more dimensions help when trying to classify the data in this dataset? How important is regularization in these cases?

Using more dimensions does help when classifing the data in this dataset. When testing all four features in the last case, we got the most accurate prediction made by the model. The Regularization was not adjusted in this case, but C was set to 100. Interestingly enough, when adjusting the L1 and L2 regularizations, the models did marginally worse at prediction accuracy. Only in the first case of Sepal length / Sepal Width did the adjusted L1 L2 regularization improve the accuracy score.

✓ 0s completed at 8:25 PM