WEEK 6 IN-CLASS ACTIVITIES/LAB

SUBMITTED BY: PRATHYUSHA HARISH KUMAR [JB24771]

TASK 1A: INTERPRETING LOGISTIC REGRESSION MODEL

Given a logistic regression model

$$\ln\left(\frac{p}{1-p}\right) = -3 + 0.8 \times \text{Hours\_Studied} + 1.5 \times \text{Review\_Session}$$

Answer the following questions:
(You may use the provided "logistic regression" notebook and AI assistant.)

a. *Thomas studied for two hours and did not attend the review session. What is his (1) log odds, (2) odds, and (3) likelihood of passing the exam?*

   SOLUTION:

```
The value of Log_odds when Review session is 0  =  -1.4
The value of odds when Review session is 0 =  0.2465969639416065
Pass_likelihood value when Review Session is 0 =  0.19781611144141825
```

b. *If Thomas goes to the review session, what is the updated 1) log_odds, (2) odds, and (3) likelihood of passing the exam?*

   SOLUTION:

```
The value of Log_odds when Review session is 1  =  0.10000000000000009
The value of odds when Review session is 1 =  1.1051709180756477
Pass_likelihood value when Review Session is 1 =  0.52497918747894
```

c. *If Thomas studied more or less hours, would the answer change?*

   SOLUTION:

   From the given equation,

   The Coefficient of the Hours_Studied = 0.8.

   Thus, with an increase in the study hours, the log odds will increase leading to the increase in the probability.

   If Thomas studies more or fewer hours, the log-odds, odds, and probability will change accordingly, meaning more study hours increase the probability of passing.

d. *How would you interpret the coefficient of review_session (1.5) from the above experiment?*

SOLUTION:

The coefficient for Review_Session (1.5) means that, when a student attends the review session, it increases the log-odds of passing by 1.5. This significantly increases the probability of passing.

e. *Using similar reasoning, how would you interpret the coefficient of hours_studied (0.8)*

SOLUTION:

The coefficient for Hours_Studied (0.8) means that for every additional hour studied, the log-odds of passing increase by 0.8, increasing the probability of passing.

f. *How would you interpret the intercept?*

SOLUTION:

The intercept (-3) represents the log-odds of passing when both Hours_Studied and Review_Session are 0.

Therefore, a student who neither studies nor attends a review session has a very low probability of passing.

g. *For someone who studied 8 hours, would you recommend him/her to attend the review session?*

SOLUTION:

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the logistic function
def logistic_function(z):
    return 1 / (1 + np.exp(-z))

# Define the model
def log_odds(hours_studied, review_session):
    return -3 + 0.8 * hours_studied + 1.5 * review_session

# Generate some data
hours_studied = 8

# Calculate log-odds and probabilities for both Review_Session=0 and Review_Session=1
log_odds_0 = log_odds(hours_studied, 0)
probability_0 = logistic_function(log_odds_0)
print('The value of Log_odds when Review session is 0  = ',log_odds_0)
print('The value of odds when Review session is 0 = ',np.exp(log_odds_0))
print('Pass_likelihood value when Review Session is 0 = ',probability_0)


log_odds_1 = log_odds(hours_studied, 1)
probability_1 = logistic_function(log_odds_1)
print('The value of Log_odds when Review session is 1  = ',log_odds_1)
print('The value of odds when Review session is 1 = ',np.exp(log_odds_1))
print('Pass_likelihood value when Review Session is 1 = ',probability_1)
```

```
The value of Log_odds when Review session is 0  =  3.4000000000000004
The value of odds when Review session is 0 =  29.964100047397025
Pass_likelihood value when Review Session is 0 =  0.9677045353015495
The value of Log_odds when Review session is 1  =  4.9
The value of odds when Review session is 1 =  134.28977968493552
Pass_likelihood value when Review Session is 1 =  0.9926084586557181
```

According to the calculated probability values, when the student does not attend the review_session the likelihood to pass is 96.7% and if the student attend the review_session, the likelihood is 99.2%.

For a student who studied 8 hours, they likely already have a high passing probability, but attending a review session would further help to boost it.

h. *What type of students seems to benefit most from the review session?*

SOLUTION:

Students who study fewer hours, benefit the most from attending the review session, as it significantly increases their probability of passing.

TASK 1B: BUILD A LOGISTIC REGRESSION MODEL

Using the dataset "student_data.csv," write code to (1) create a visualization of the data, (2) fit a model using logistic regression, (3) output model coefficients and performance metrics such as accuracy and AUC and ROC; ***NOTE: For this exercise, you will train and test on the same given dataset, instead of doing train/test split. Make sure you give the correct GPT prompt.***

SOLUTION:

CW3 LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve, confusion_matrix
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files student_data.csv

- **student_data.csv**(text/csv) - 2275 bytes, last modified: 3/10/2025 - 100% done

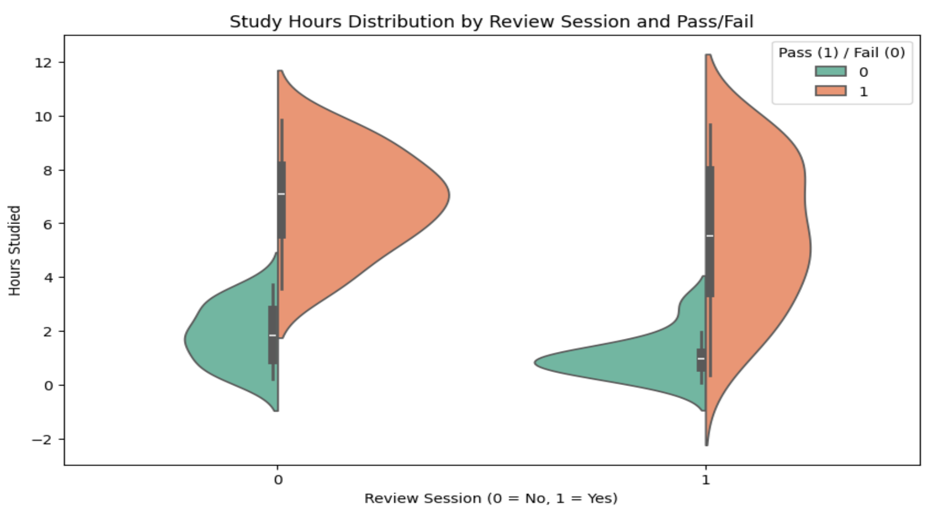
```
Saving student_data.csv to student_data.csv
```

```
# Load the dataset
data = pd.read_csv('student_data.csv')

# Display basic information about the dataset
display(data.head())
print(data.info())
```

	Hours_Studied	Review_Session	Results
0	3.745401	0	0
1	9.507143	1	1
2	7.319939	0	1
3	5.986585	0	1
4	1.560186	1	1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Hours_Studied   100 non-null    float64
 1   Review_Session  100 non-null    int64
 2   Results         100 non-null    int64
dtypes: float64(1), int64(2)
memory usage: 2.5 KB
None
```

```
# Data Visualization
plt.figure(figsize=(10, 6))
sns.violinplot(data=data, x='Review_Session', y='Hours_Studied', hue='Results', split=True, palette='Set2')
plt.title('Study Hours Distribution by Review Session and Pass/Fail')
plt.xlabel('Review Session (0 = No, 1 = Yes)')
plt.ylabel('Hours Studied')
plt.legend(title='Pass (1) / Fail (0)')
plt.show()
```




```
# Logistic Regression Model
X = data[['Hours_Studied', 'Review_Session']]
y = data['Results']
```

```
model = LogisticRegression(solver='liblinear', C=1.0, max_iter=200, random_state=42)
model.fit(X, y)
```

Commands | + Code + Text

```
model = LogisticRegression(solver='liblinear', C=1.0, max_iter=200, random_state=42)
model.fit(X, y)
```

```
LogisticRegression
LogisticRegression(max_iter=200, random_state=42, solver='liblinear')
```

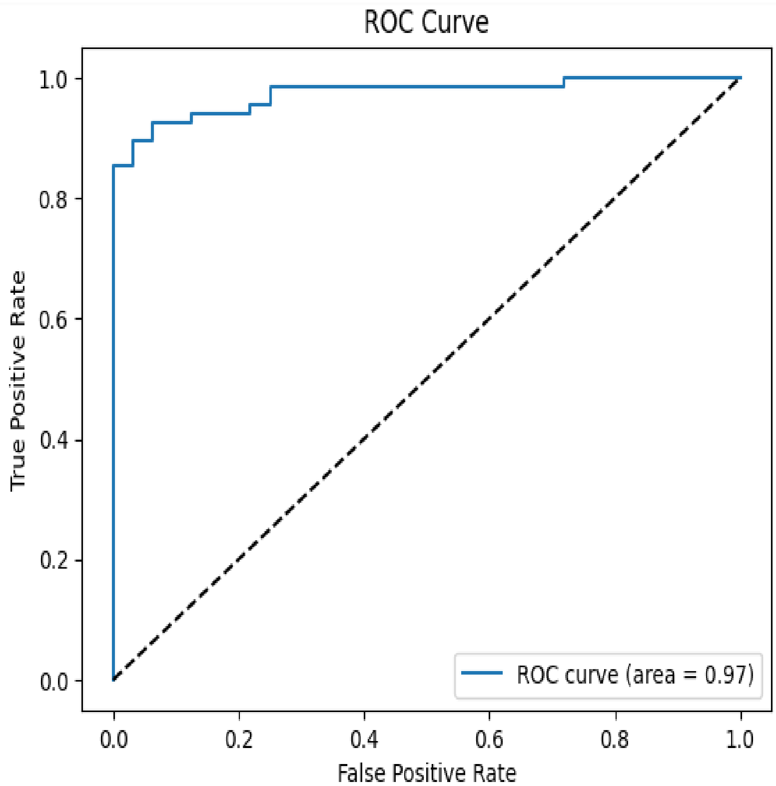
```
# Model Coefficients
print('Intercept:', model.intercept_)
print('Coefficients:', model.coef_)
```

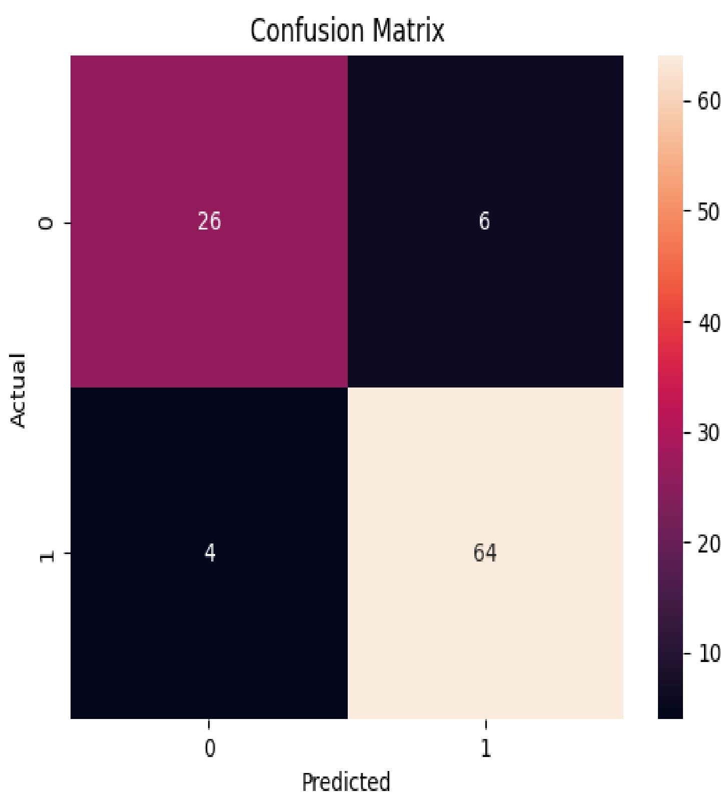
```
Intercept: [-2.77870623]
Coefficients: [[0.92519784 1.10466804]]
```

```
# Model Predictions
predictions = model.predict(X)

# Accuracy

accuracy = accuracy_score(y, predictions)
print('Accuracy:', accuracy*100, '%')
```

```
Accuracy: 90.0 %
```

```
# ROC and AUC
probabilities = model.predict_proba(X)[:, 1]
auc = roc_auc_score(y, probabilities)
print('AUC:', auc)

fpr, tpr, _ = roc_curve(y, probabilities)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()

# Confusion Matrix
cm = confusion_matrix(y, predictions)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```


ROC Curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve (area = 0.97)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate


Confusion Matrix
26
6
4
64
0
1
Actual
0
1
Predicted
60
50
40
30
20
10

TASK 2: UNDERSTANDING AND PREVENT OVERFITTING IN THE CONTEXT OF SVM

Write code to fit a Support Vector Machine model using (1) linear kernel and (2) RBF kernel. For the RBF kernel, use grid search to find the best gamma parameter using k-fold cross-validation.

SOLUTION:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve, confusion_matrix
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files student_data.csv
• student_data.csv(text/csv) - 2275 bytes, last modified: 3/10/2025 - 100% done
Saving student_data.csv to student_data.csv
```

```
# Load the dataset
data = pd.read_csv('student_data.csv')

# Display basic information about the dataset
display(data.head())
print(data.info())
```

	Hours_Studied	Review_Session	Results
0	3.745401	0	0
1	9.507143	1	1
2	7.319939	0	1
3	5.986585	0	1
4	1.560186	1	1

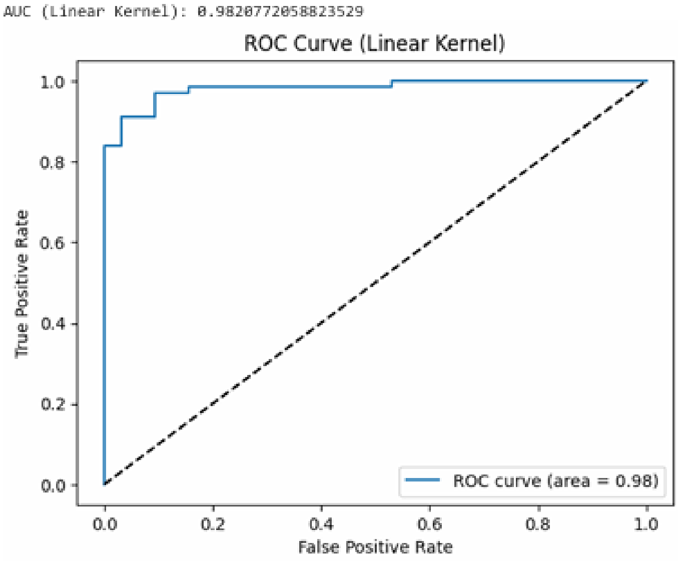
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Hours_Studied   100 non-null    float64
 1   Review_Session  100 non-null    int64
 2   Results         100 non-null    int64
dtypes: float64(1), int64(2)
memory usage: 2.5 KB
None
```

```
# Prepare the features and target variable
X = data[['Hours_Studied', 'Review_Session']]
y = data['Results']

# SVM with Linear Kernel
linear_svm = SVC(kernel='linear', probability=True, random_state=42)
linear_svm.fit(X, y)

# Predictions and Accuracy (Linear Kernel)
predictions_linear = linear_svm.predict(X)
accuracy_linear = accuracy_score(y, predictions_linear)
print('Accuracy (Linear Kernel):', accuracy_linear * 100, '%')
```

```
Accuracy (Linear Kernel): 92.0 %
```

```
# AUC and ROC (Linear Kernel)
probabilities_linear = linear_svm.predict_proba(X)[:, 1]
auc_linear = roc_auc_score(y, probabilities_linear)
print('AUC (Linear Kernel):', auc_linear)

fpr, tpr, _ = roc_curve(y, probabilities_linear)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_linear)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve (Linear Kernel)')
plt.legend(loc='lower right')
plt.show()
```

AUC (Linear Kernel): 0.9820772058823529




```
# RBF Kernel with Grid Search for Best Gamma
param_grid = {'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}
rbf_svm = SVC(kernel='rbf', probability=True, random_state=42)
grid_search = GridSearchCV(rbf_svm, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X, y)
```

GridSearchCV

best_estimator_: SVC

SVC(gamma=0.1, probability=True, random_state=42)

SVC

```
# Best Gamma
best_gamma = grid_search.best_params_['gamma']
print('Best Gamma:', best_gamma)
```

Best Gamma: 0.1

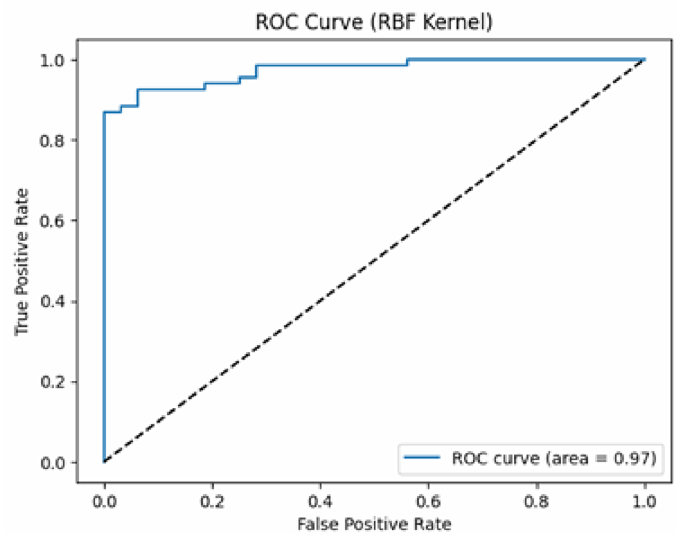
```
# Train SVM with RBF Kernel using Best Gamma
best_rbf_svm = SVC(kernel='rbf', gamma=best_gamma, probability=True, random_state=42)
best_rbf_svm.fit(X, y)

# Predictions and Accuracy (RBF Kernel)
predictions_rbf = best_rbf_svm.predict(X)
accuracy_rbf = accuracy_score(y, predictions_rbf)
print('Accuracy (RBF Kernel):', accuracy_rbf * 100, '%')
```

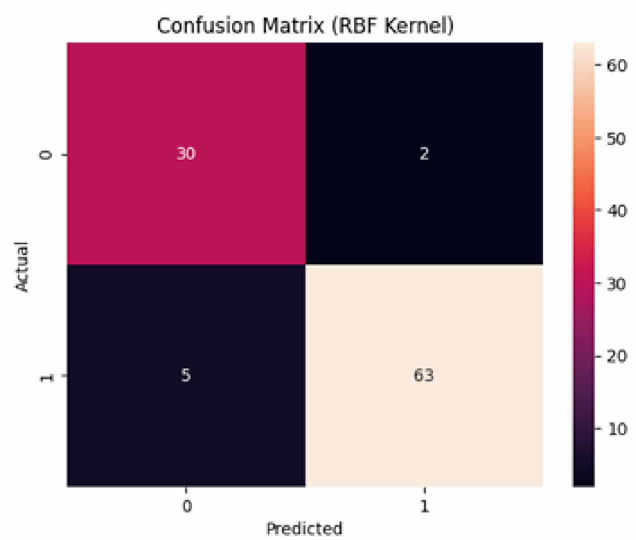
Accuracy (RBF Kernel): 93.0 %

```
# AUC and ROC (RBF Kernel)
probabilities_rbf = best_rbf_svm.predict_proba(X)[:, 1]
auc_rbf = roc_auc_score(y, probabilities_rbf)
print('AUC (RBF Kernel):', auc_rbf)

fpr, tpr, _ = roc_curve(y, probabilities_rbf)
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_rbf)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve (RBF Kernel)')
plt.legend(loc='lower right')
plt.show()
```

```
cm = confusion_matrix(y, predictions_rbf)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix (RBF Kernel)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
AUC (RBF Kernel): 0.9738051470588235
```






```
# External Input Prediction for svm with rbf kernel
def predict_pass(hours, review):
    input_data = pd.DataFrame([[hours, review]], columns=['Hours_Studied', 'Review_Session'])
    prediction = best_rbf_svm.predict(input_data)
    probability = best_rbf_svm.predict_proba(input_data)[0, 1] * 100
    print(f'Predicted Result: {prediction[0]}, Probability: {probability:.2f}%')

# Example of external input
predict_pass(5, 0)  # Predicting for 5 study hours and attending the review session
```

```
Predicted Result: 1, Probability: 95.89%
```

```
# External Input Prediction for svm with rbf kernel
def predict_pass(hours, review):
    input_data = pd.DataFrame([[hours, review]], columns=['Hours_Studied', 'Review_Session'])
    prediction = linear_svm.predict(input_data)
    probability = linear_svm.predict_proba(input_data)[0, 1] * 100
    print(f'Predicted Result: {prediction[0]}, Probability: {probability:.2f}%')
```

```
# Example of external input
predict_pass(5, 0)  # Predicting for 5 study hours and attending the review session
```

```
Predicted Result: 1, Probability: 80.82%
```