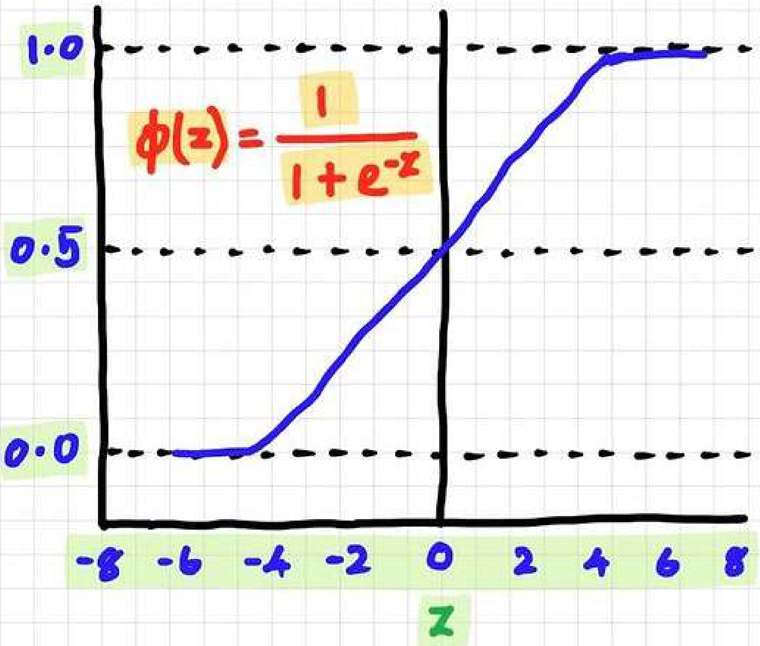
COMPLETE
LOGISTIC REGRESSION CHEATSHEET
ALONG WITH SOURCE CODE



WHAT IS LOGISTIC REGRESSION?

- Logistic Regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). In other words, it predicts the probability of occurrence of an event by fitting data to a logit function.

COMPLETE

LOGISTIC REGRESSION CHEATSHEET

ALONG WITH SOURCE CODE

USECASE:

- It's widely used for binary classification problems, for example:
    - Predicting if a student passes or fails based on hours of studying.
    - Determining if a transaction is fradulent or not.

ADVANTAGES OF LOGISTIC REGRESSION:

- Simple and linear.
- Requires less training time.
- It's interpretable.

DISADVANTAGES OF LOGISTIC REGRESSION:

- Assumes linear decision boundry.
- Not powerful enough to capture complex relationships.

COMPLETE
LOGISTIC REGRESSION CHEATSHEET
ALONG WITH SOURCE CODE

HYPERPARAMETER IN LOGISTIC REGRESSION :

- C (Inverse of regularization strength): Smaller values specify stronger regularization. Regularization can help prevent overfitting.

- Penalty (used to specify the norm used in the penalization): options are 'L1', 'L2', 'elasticnet', or 'none'. L1 and L2 are the most common.

- Solver (Algorithm to use in the optimization problem): Algorithms like 'newton-cg', 'lbfgs', 'liblinear', 'sag', and 'saga' can be used. The default is 'lbfgs'.

- max_iter (Maximum number of iterations for solvers to converge): The default is 100.

- fit_intercept (specifies if a constant should be added to the decision function): It's a Boolean value.

- multi_class (Algorithms to use for multiclass problems): options include 'auto', 'ovr', 'multinomial'.

COMPLETE
LOGISTIC REGRESSION CHEATSHEET
ALONG WITH SOURCE CODE

PYTHON CODE TO IMPLEMENT LOGISTIC REGRESSION:

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split,
                                     GridSearchCV
from sklearn.metrics import classification_report

#Load dataset
data = load_iris()
x = data.data
y = data.target

#Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(x, y,
                                         test_size=0.25,
                                         random_state=0)

#Initialize Logistic Regression
clf = LogisticRegression()
```

COMPLETE

LOGISTIC REGRESSION CHEATSHEET

ALONG WITH SOURCE CODE

```
# Define hyperparameters grid for tuning
param_grid = {
    'C': np.logspace(-4, 4, 20),
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [100, 500, 1000, 1500],
}

# Use GridSearchCV to find the best hyperparameters
grid_search = GridSearchCV(clf, param_grid, cv=5, verbose=1,
                           n_jobs=-1)
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print("Best Parameters:", grid_search.best_params_)

# Use the best model to predict the test data
best_clf = grid_search.best_estimator_
predictions = best_clf.predict(X_test)

# Print the classification report
print(classification_report(y_test, predictions))
```

COMPLETE
LOGISTIC REGRESSION CHEATSHEET
ALONG WITH SOURCE CODE

THINGS TO NOTE:

1. The param_grid in GridSearchCV specifies the different hyperparameters we want to try and their potential values.

2. CV=5 means that a 5-fold cross-validation is used on the train dataset to evaluate the hyperparameters.

3. n_jobs=-1 allows the process to use all available CPUs for faster computation.

4. Before deploying this code, you might want to ensure you've installed necessary libraries and adjust the parameters of param_grid as per your requirements.