Multiple Regression Model

Imports

```
%%capture
%pip install -qqqqqq -U -q -g -e .[dev,llm,sae,mamba,hf-kernels,hf-xet]
%pip install -qqqqqq -U -q -g -e git+https://github.com/jaxtyping-group/jaxtyping
%pip install -qqqqqq -U -q -g -e git+https://github.com/beartype/beartype
%pip install -qqqqqq -U -q -g -e git+https://github.com/TransformerLensOrg/TransformerLens
%pip install -qqqqqq -U -q -g -e git+https://github.com/jaxtyping-group/tensor-jaxtyping
```

```
jaxtyping 0.3.4.dev0+cu117
```

```
# ruff: noqa: E402, F401, F403, F405, E731, E741, E501, E203, E266, E221, E222, E241, E251, E272, E275, E302, E305, E306, E703, E704, E713, E714, E721, E722, E723, E731, E741, W291, W293, W391, W503, W504, W605, W606, W611, W612, W613, W614, W615, W616, W617, W618, W619, W620, W621, W622, W623, W624, W625, W626, W627, W628, W629, W630, W631, W632, W633, W634, W635, W636, W637, W638, W639, W640, W641, W642, W643, W644, W645, W646, W647, W648, W649, W650,
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
model = Sequential([
    Dense(512, input_shape=(784,)),
    Activation('relu'),
    Dropout(0.2),
    Dense(512),
    Activation('relu'),
    Dropout(0.2),
    Dense(10),
    Activation('softmax')
])
```

```
from keras.models import Sequential
model = Sequential()
model.add(Dense(512,
                input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
```

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
model.fit(x_train, y_train,
          batch_size=128,
          epochs=20,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.08481131
Test accuracy: 0.9829
```

Is it possible implement a Regression learning model using logistic regression model

Logistic regression is a classification algorithm, not a regression algorithm. It is used for predicting binary outcomes (e.g. 0 or 1, yes or no) based on a set of input features. It aims to apply a sigmoid function to map the output of a linear combination of the input features to a probability between 0 and 1.

In contrast, regression algorithms are used for predicting a continuous numerical value (e.g. the price of a house, the weight of a person) based on a set of input features. Some examples of regression algorithms include linear regression, polynomial regression, and support vector regression.

So, it is not possible to implement a regression learning model using the logistic regression model. However, you can use logistic regression to perform binary classification (e.g. predicting whether a patient has a disease or not based on their symptoms) and then use the predicted probabilities from the logistic regression model (e.g. predicting the severity of the disease based on the patients' lab test and other features).

One vs Rest and One vs One

```
# Import the necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
```

```
# Load the iris dataset
iris = load_iris()
X, y = iris.data, iris.target
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
# Create a multi-class classifier using the one-vs-rest strategy
clf = OneVsRestClassifier(SVC())
```

```
# Fit the classifier to the training data
clf.fit(X_train, y_train)
```

```
OneVsRestClassifier(estimator=SVC())
```

```
# Predict the labels for the test data
y_pred = clf.predict(X_test)
```

```
# Import the necessary libraries
from sklearn.metrics import accuracy_score
```

```
# Calculate the accuracy of the classifier
accuracy = accuracy_score(y, y_pred)
print(accuracy)
```

```
0.9733
```

```
# Import the iris dataset and the necessary libraries
from sklearn.datasets import load_iris
```

```
# Load the iris dataset
iris = load_iris()
```

```
# Define the features and the target variable
X = iris.data
y = iris.target
```

```
# Import the OneVsOneClassifier and the SVC classifier
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC
```

```
# Create a multi-class classifier using the one-vs-one strategy
clf = OneVsOneClassifier(SVC())
```

```
# Fit the classifier to the training data
clf.fit(X, y)
```

```
OneVsOneClassifier(estimator=SVC())
```

```
# Predict the labels for the test data
y_pred = clf.predict(X)
```

```
# Import the accuracy_score function
from sklearn.metrics import accuracy_score
```

```
# Calculate the accuracy of the classifier
accuracy = accuracy_score(y, y_pred)
print(accuracy)
```

```
0.97
```