In [ ]:
```
import math
import json
import random
import numpy as np
import pandas as pd
```

In [ ]:
```
# Read dataset
glove_df_train = pd.read_csv("../project_data/data/glove/glove.train.csv")
glove_df_test = pd.read_csv("../project_data/data/glove/glove.test.csv")
glove_df_eval = pd.read_csv("../project_data/data/glove/glove.eval.anon.csv"

misc_df_train = pd.read_csv("../project_data/data/misc/misc-attributes-trair
misc_df_test = pd.read_csv("../project_data/data/misc/misc-attributes-test.c
misc_df_eval = pd.read_csv("../project_data/data/misc/misc-attributes-eval.c

# Add label to misc attribute
misc_df_train["label"] = glove_df_train["label"]
misc_df_test["label"] = glove_df_test["label"]
misc_df_eval["label"] = glove_df_eval["label"]

# Add bias
glove_df_train["bias"] = 1
glove_df_test["bias"] = 1
glove_df_eval["bias"] = 1

glove_df_eval
```

Out[ ]:

	label	x0	x1	x2	x3	x4	x5	
0	1	-4.400295	4.717408	-7.981161	0.582802	6.156548	0.865143	3.9
1	1	-4.358865	-2.167632	-7.009697	2.813710	13.745421	-1.438060	5.2
2	1	-5.584966	0.501010	-1.244940	2.081082	7.261350	-1.760596	-1.
3	1	-13.807071	-2.762292	-15.260910	3.593135	10.570365	-1.137067	4.
4	1	-13.159473	-6.476247	-9.394270	-0.055009	17.871582	4.610767	5.5
...	...	...	...	...	...	...	...	
5245	1	-1.847650	-0.982201	-3.223992	0.838675	5.136714	-4.103040	-0.5
5246	1	-3.905437	-2.460167	-3.065652	-2.345944	3.024367	-1.337712	0.9
5247	1	-0.558323	-1.632193	-1.525511	-1.881319	4.102118	-2.785541	4.
5248	1	-7.203439	-2.136733	-8.628859	-0.759335	5.817169	3.166624	2.4
5249	1	-7.297932	-0.127681	-5.943969	1.595681	2.993623	-4.303490	-1.

5250 rows × 302 columns

In [ ]:
```python
def get_max_key_by_value(map):
    max_key = ""
    max_val = float("-inf")

    for key, val in map.items():
        if val > max_val:
            max_val = val
            max_key = key

    # print("map: ", map, "max_key: ", max_key)
    return max_key

def initialize_weights_bias(rand_start, rand_end, feature_count):
    random_number = random.uniform(rand_start, rand_end)

    bias = random_number
    weights = []  # All weights and bias should be same.
    for _ in range(feature_count):
        weights.append(random_number)

    return weights, bias

def predict(example, weights):
    value = np.dot(weights, example)
    return 1 if value > 0 else -1

def test_accuracy(df, weights, store_eval=False):
    total = df.shape[0]
    correct_prediction = 0
    eval_list = []

    for _, row in df.iterrows():
        example = row.tolist()
        actual_label = example[0]  # y
        example = example[1:]  # x

        predicted_label = predict(example, weights)

        if store_eval:
            eval_list.append(predicted_label)

        if predicted_label == actual_label:
            correct_prediction += 1

    # print(f"Test accuracy. Correct Pred: {correct_prediction}, Total: {tot
    return correct_prediction / total, eval_list

def perceptron(df, learning_rate, weights):
    update_count = 0
    for _, row in df.iterrows():
        example = row.tolist()
        actual_label = example[0]  # y
        example = example[1:]  # x
        value = actual_label * (np.dot(weights, example))

        # update
        if value < 0:
            update_count += 1
            for index in range(len(weights)):
                # w = w + r * y * x
                weights[index] += learning_rate * actual_label * example[inc

    return weights, update_count
```

```
In [ ]: rand_start = -0.01
        rand_end = 0.01

        initial_weights, _ = initialize_weights_bias(
            rand_start=rand_start, rand_end=rand_end, feature_count=glove_df_train.s
        )

        accuracy_dict = {}
        prediction_list_dict = {}
        learning_rates = [1, 0.1, 0.01]

        for learning_rate in learning_rates:
            weights, _ = perceptron(df=glove_df_train, learning_rate=learning_rate,

            accuracy, prediction_list = test_accuracy(df=glove_df_eval, weights=weig
            accuracy_dict[accuracy] = accuracy
            prediction_list_dict[accuracy] = prediction_list

        print(f"Accuracy of tree on eval dataset: ", get_max_key_by_value(accuracy_c
        df = pd.DataFrame(prediction_list_dict[accuracy])
        df.to_csv("perceptron_glove_eval_dataset_prediction.csv", index=True, header
```

```
Accuracy of tree on eval dataset:  1.0
```