به نام خدا

دانشکده مهندسی برق

گزارش درس یادگیری ماشین

مقطع: کارشناسی ارشد گرایش: مهندسی کنترل

گزارش مینی پروژه اول

توسط:

مرجان محمدی

۴۰۱۱۱۵۳۴

استاد درس:

دکتر علیاری

بهار ۱۴۰۳

فهرست مطالب

سوال اول.....۳

سوال دوم.....۱۴

سوال سوم.....۱۶

سوال اول)

برای توضیح روند آموزش و ارزشیابی مدل طبقه بندی خطی و نشان دادن آن در قالب بلوک دیاگرام، ابتدا به ترسیم و توضیح نمودار می پردازیم و سپس نحوه تغییر نوع طبقه بندی از دو کلاسه به چند کلاسه را توضیح می دهیم. نمودار را تحت تاثیر قرار می دهد.

بلوک دیاگرام مدل طبقه بندی خطی

بلوک دیاگرام که ما طراحی می کنیم شامل اجزای زیر است:

INPUT DATA: حاوی ویژگی ها و برچسب های مربوط به داده ها است.

پیش پردازش داده: شامل مقیاس بندی ویژگی ها، پر کردن داده های از دست رفته و سایر تکنیک های پاکسازی داده ها.

تقسیم داده ها: تقسیم داده ها به دو مجموعه آموزش و آزمایش.

آموزش مدل: استفاده از داده های آموزشی برای آموزش مدل طبقه بندی کننده.

مدل طبقه بندی کننده خطی: مدلی که برای طبقه بندی داده ها با استفاده از الگوریتم هایی مانند رگرسیون لجستیک یا ماشین بردار پشتیبان استفاده می شود.

ارزیابی مدل: استفاده از داده های آزمون برای ارزیابی کارایی مدل.

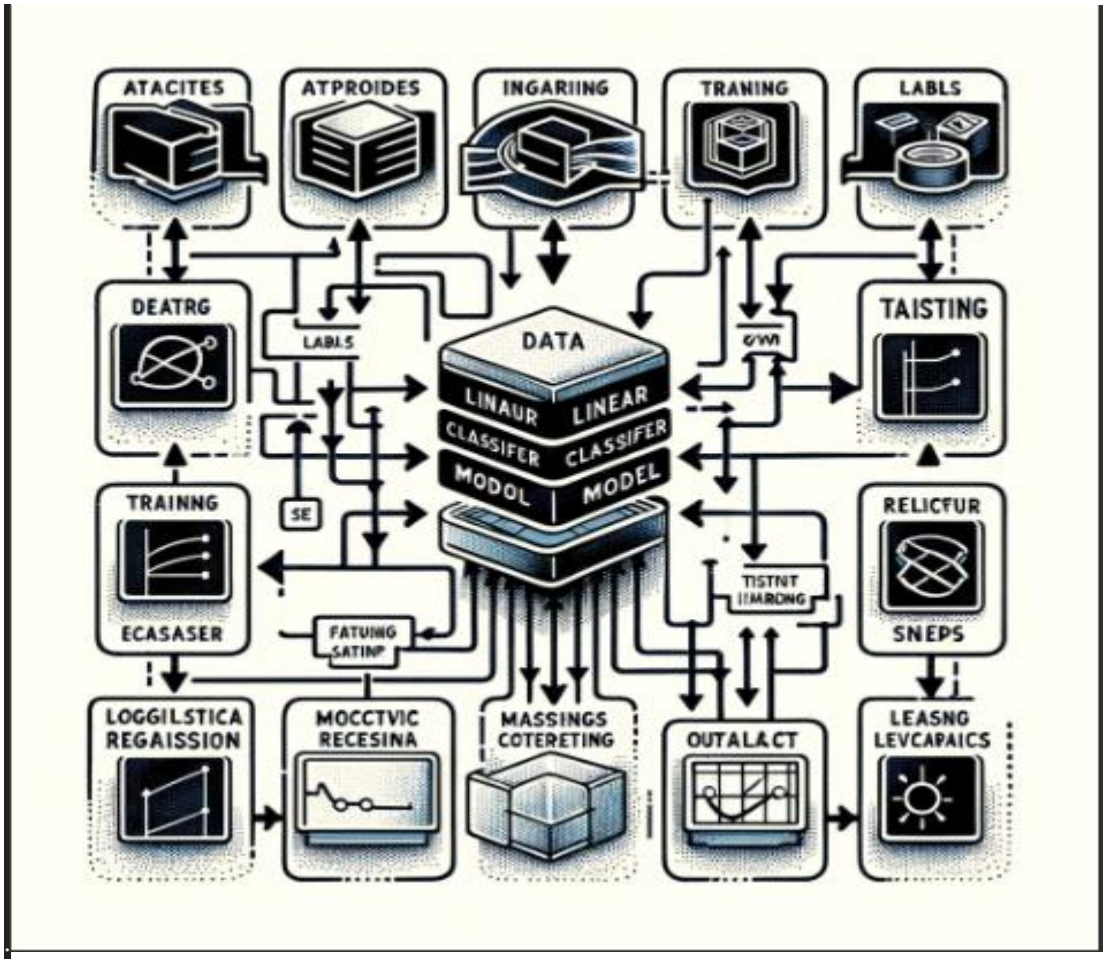
خروجی: پیش بینی کلاس برای نمایش داده های جدید یا دقت مدل.

تغییر نوع طبقه بندی از دو کلاسه به چند کلاسه

تغییر نوع طبقه بندی از دو کلاسه به چند کلاسه بیشترین تاثیر را در بلوک Linear Classifier Modelایجاد می کند. در حالت دو کلاسه، الگوریتم هایی مانند رگرسیون لجستیک تنها دو کلاس را تشخیص می دهند و خروجی باینری است (مثلاً ۰ و ۱). اما برای حالت چند کلاسه، مدل باید بتواند بین چند کلاس مختلف تمایز قائل شود. این معمولاً از طریق

استراتژی هایی مانند «یک در مقابل همه» یا «یک در مقابل یک» در رگرسیون لجستیک یا عملکردی انجام می شود.

این کار در ماشین های بردار پشتیبان انجام می شود.



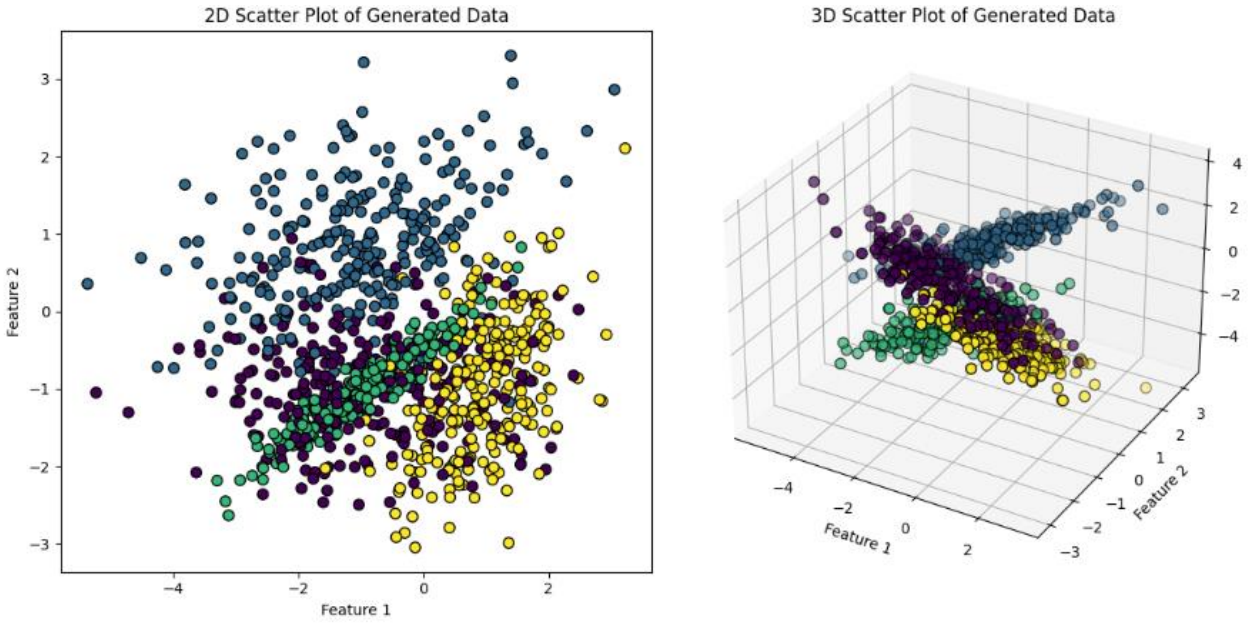
شکل۱: دیاگرام بلوکی فرآیند آموزش یک مدل طبقه بند خطی

دیاگرام بلوکی مدل طبقه‌بند خطی که شامل اجزای مورد نظرما است، در شکل ۱ تهیه شده است. این دیاگرام شامل مراحل داده‌های ورودی، پیش‌پردازش‌ها، تقسیم‌بندی داده‌ها، آموزش مدل، مدل طبقه‌بند خطی، ارزیابی مدل و خروجی است. هر بخش به وضوح مشخص شده و جریان‌ها از یک مرحله به مرحله دیگر با فلش‌ها نمایش داده شده است.

در ابتدا با دستور make classification و ویژگی های تعریف شده یک دیتاست جنریت می کنیم.

```python
def generate_dataset():
    """Generate a synthetic classification dataset."""
    X, y = make_classification(
        n_samples=1000,
        n_features=3,
        n_classes=4,
        n_informative=3,
        n_redundant=0,
        n_repeated=0,
        n_clusters_per_class=1,
        class_sep=1,
        random_state=69
    )
    return X, y
```

دیتاست تولید شده ما در شکل ۲ نمایش داده شده است.



شکل۲: دیتاست تولید شده

با توجه به شکل موجود، می‌توانیم طبقه‌بندی‌هایی را برای این دیتاست مناسب دریافت کنیم؛ زیرا داده‌ها به خوبی از سایر کلاس‌ها جدا نشده و نقاط مربوط به کلاس‌های مختلف به صورت خطی قابل تمیز کردن است. زیر استفاده کرد:

کاهش ویژگی :این ویژگی میزان جداسازی و فاصله بین داده ها را نشان می دهد. هر چه میزان این ویژگی کمتر باشد، جداسازی داده‌ها خواهد بود.

ایجاد عدم تعادل در داده‌ها: با تغییر نسبت داده‌ها، می‌توان هر کلاس را برهم زد. این اقدام از طریق ویژگی وزن می‌شود و ممکن است اشتباه توسط ماشین را افزایش دهد.

اعمال نویز به داده‌ها: با افزودن نویز به داده‌ها، می‌توان در بررسی ماشین ایجاد کرد و دقت آن را کاهش داد.

افزایش استقلال خطی بین ویژگی‌ها: ویژگی n_informative تعیین می‌کند که چه تعداد از ویژگی‌ها باید استقلال خطی باشند. در دیتاست موجود، n_informative برابر با ۳ در نظر گرفته شده است. اگر این مقدار کاهش یابد و به جای آن، ویژگی‌های n_redundant را افزایش دهد، دیتاست راحت‌تر قابل تفکیک خواهد بود.

در اولین قدم دیتای ایجاد شده را به دو قسمت ترین و ولیدیشن تقسیم می کنیم و ۸۰ درصد دیتا برای فرآیند آموزش و ۲۰ درصد دیتا برای فرآیند ولیدیشن در نظر گرفته شده است.

در قدم بعدی دیتا باید نرمالیزه شود.(به دلیل اینکه ارزش کل دیتا ها را برای ماشین مشخص کنیم همه را بین دو عدد مشخص نرمالایز می کنیم.)

روش کارکرد Standard Scaler

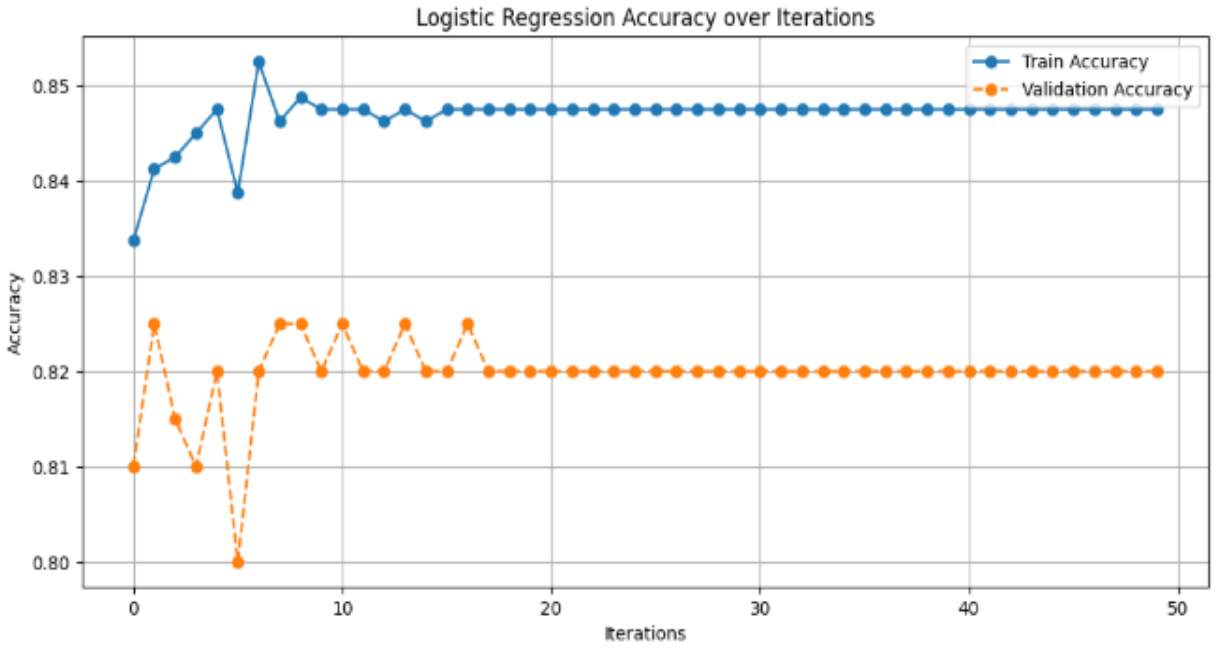
این تقسیم بندی با تغییر هر ویژگی (متغیر) به گونه‌ای است که آن صفر و انحراف آن یک، انجام می‌شود. روش محاسبهبه گونه ای است که ابتدا میانگین هر ویژگی محاسبه می‌شود. سپس انحراف معیار هر ویژگی اندازه گیری می شود، در نتیجه ویژگی ها از میانگین کم شده و بر انحراف معیار تقسیم می شوند.

سپس ۴ مدل با خصوصیات زیر به صورت زیر پیاده سازی می شود:

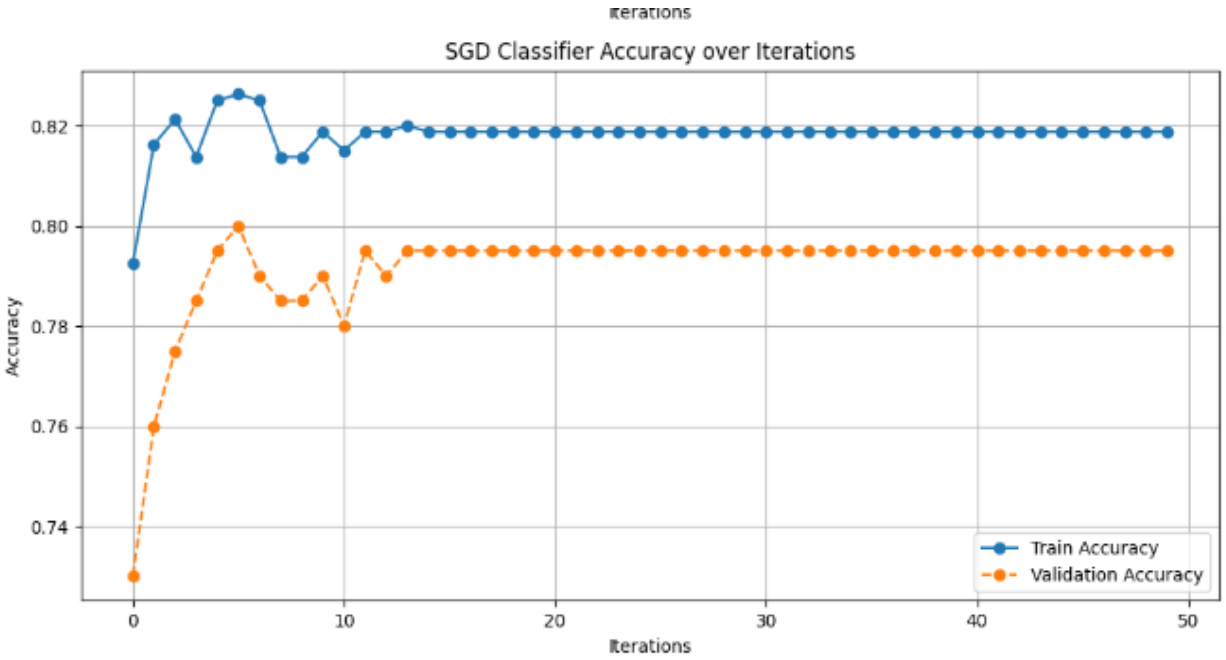
```
from sklearn.linear_model import
LogisticRegression, SGDClassifier, Perceptron,
PassiveAggressiveClassifier


models = {
        'Logistic Regression':
LogisticRegression(solver='sag',
random_state=69),
```

```
        'SGD Classifier':
SGDClassifier(max_iter=1,
learning_rate='constant', eta0=0.01,
alpha=0.0001, random_state=69),
        'Perceptron': Perceptron(max_iter=1,
eta0=1.0, random_state=69),
        'Passive Aggressive':
PassiveAggressiveClassifier(max_iter=1,
random_state=69)
    }
```

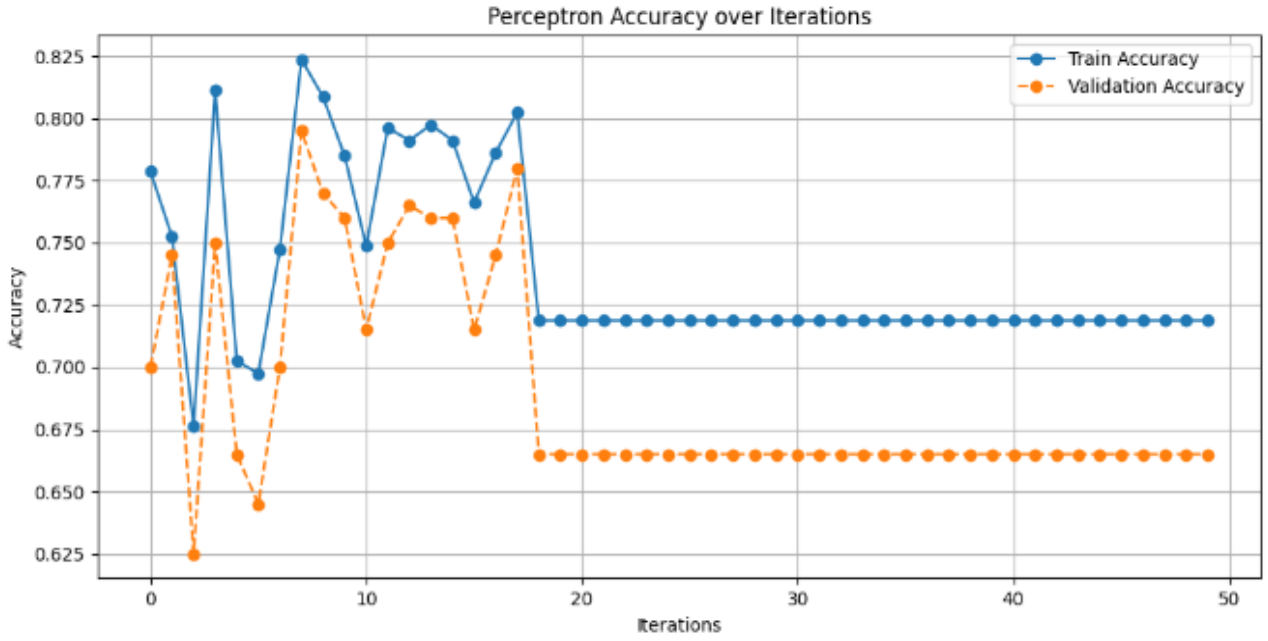
در نتیجه پاسخ های هر ۴ مدل به ترتیب در زیر آورده شده است:



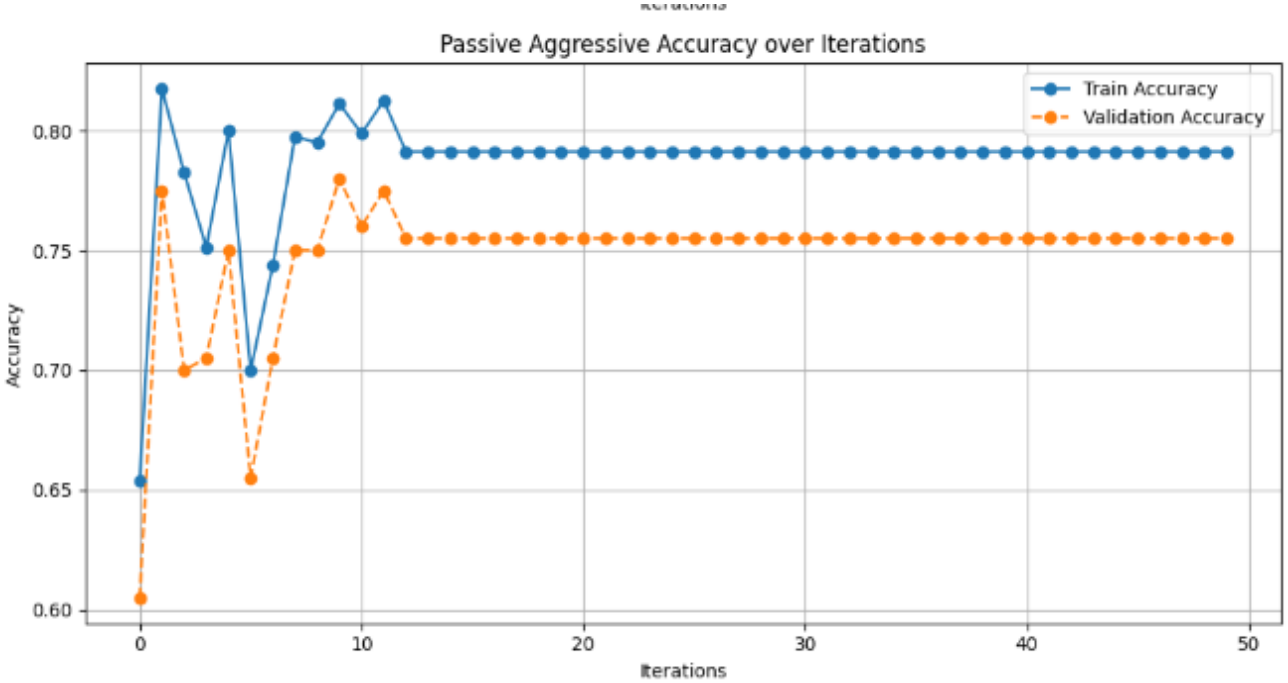
شکل۳: پاسخ مدل اول



شکل۴: پاسخ مدل دوم



شکل۵: پاسخ مدل سوم



شکل۶: پاسخ مدل چهارم

اصلی‌ترین روش پیشنهادی برای بهبود مدل، نرم‌افزارهای داده‌شده ورودی است که شامل تغییر داده‌ها می‌شود - ۱ تا ۱ می‌باشد تا بر همه ویژگی‌ها یکسان شود.

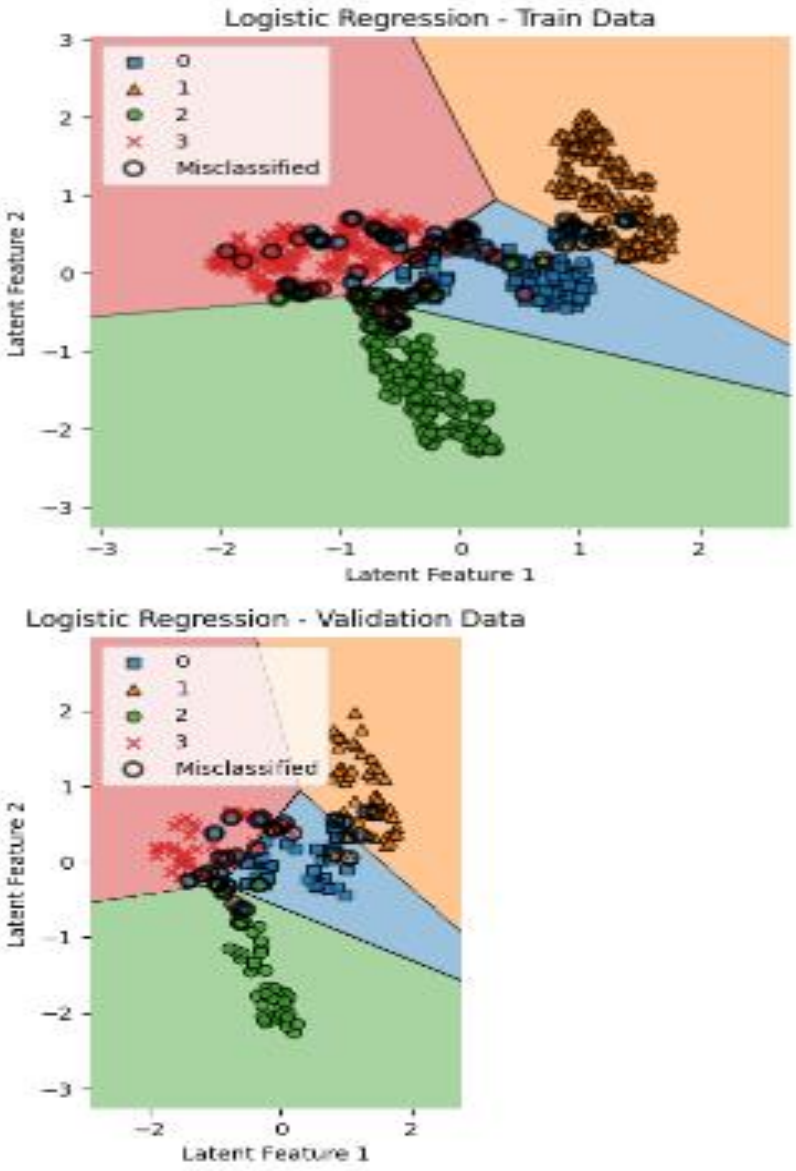
برای رفع عدم تعادل در داده‌ها و جلوگیری از سوگیری مدل به سمت کلاس‌های با داده‌های بیشتر، تکنیک‌های ویژگی به کار می‌روند. از توابع مناسب و استفاده از الگوریتم بهینه سازی مناسب برای افزایش دقت مدل کمک می‌کند.

میزان داده‌های آموزشی و تعداد ویژگی‌ها بر مدل پیچیدگی تأثیر می‌گذارند و باید متناسب با مدل‌سازی باشند. از Regularization برای کاهش واریانس و بهبود عملکرد مدل استفاده می‌شود.

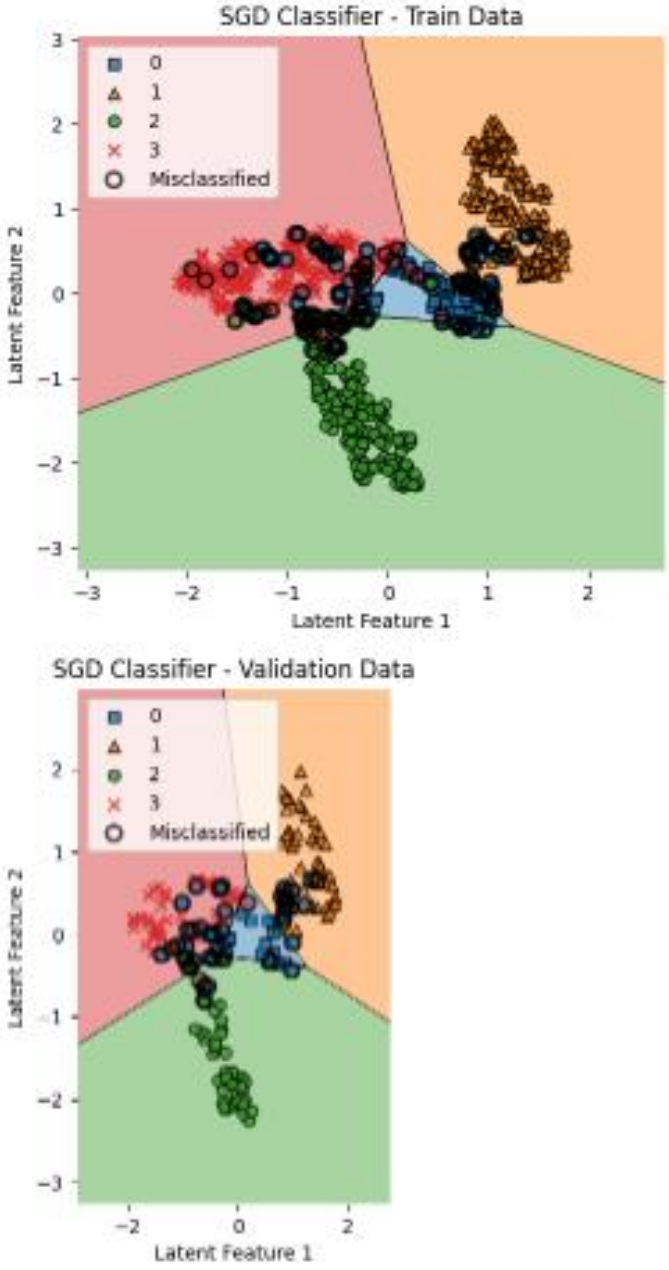
ما از روش نرمالایز کردن و آموزش دیتا روی مدل های مختلف استفاده کردیم.

سپس دیتاها را که دارای ۳ ویژگی هستند توسط t-sne به دو ویژگی کاهش می دهیم.

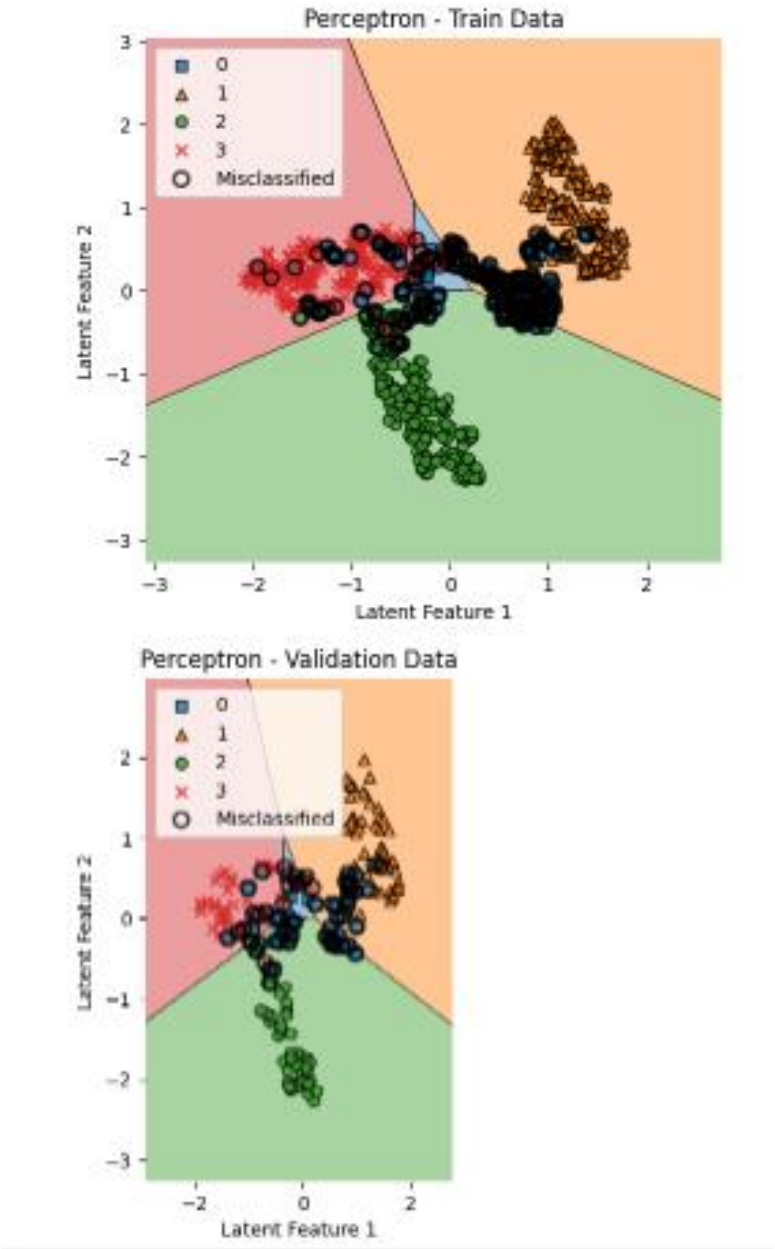
سپس مراحل قبل را روی دیتاها انجام داده و با هر ۴ مدل آموزش می دهیم و نواحی را در آخر رسم می کنیم.



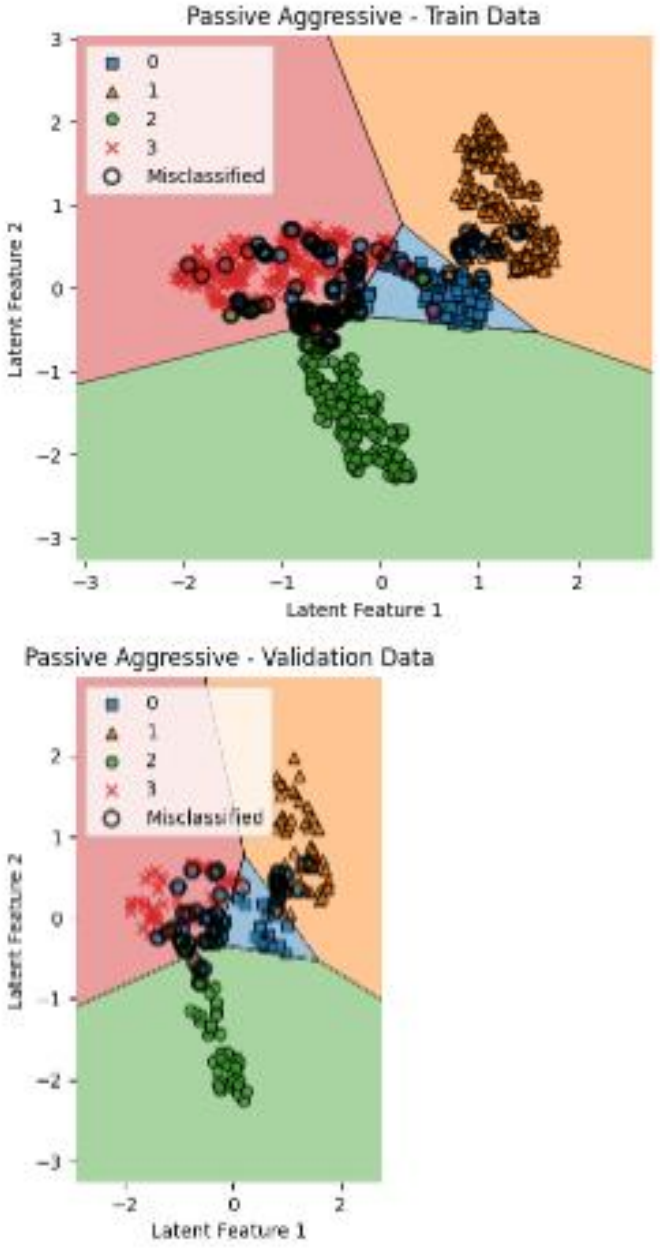
شکل۷: نواحی مدل اول



شکل۸: نواحی مدل دوم

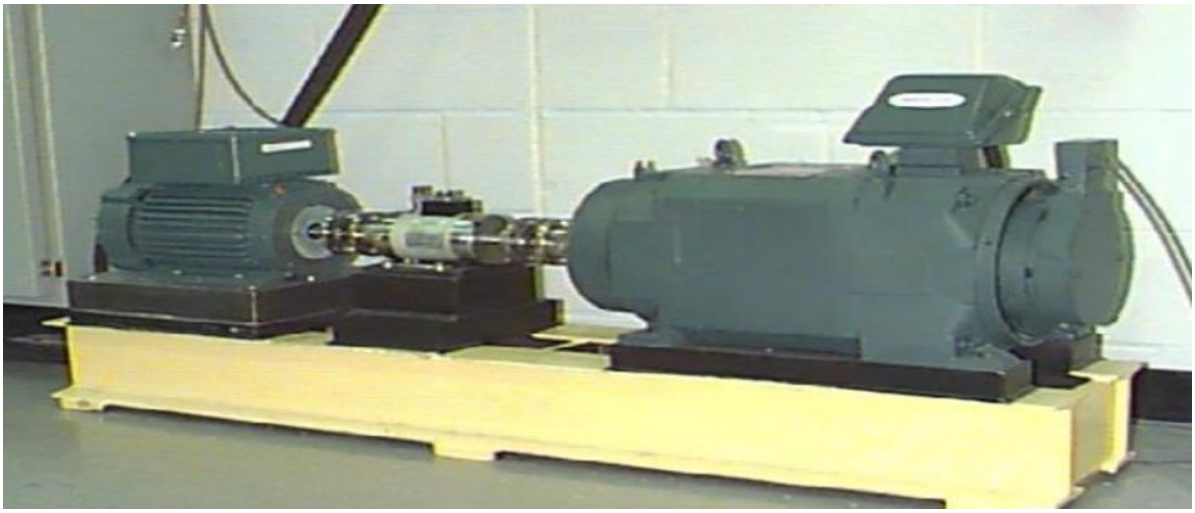


شکل۹: نواحی مدل سوم



شکل ۱۰: نواحی مدل چهارم

سوال دوم)



شکل۱۱: سیستم مورد آزمایش

این داده‌ها اطلاعات بلبرینگ‌های سالم و معیوب را می‌سازند. آزمایش‌ها با استفاده از موتور Reliance Electric دو اسب بخار انجام شده و داده‌های ارتعاشی از نقاط مختلف موتورهای اندازه‌گیری شده است. در جریان این آزمایش‌ها، شرایط واقعی موتور و خطاهای بلبرینگ به دقت ثبت شدند. بلبرینگهای معیوب با استفاده از ماشینکاری الکتریکی ((EDMو با خطاهای مختلف ساخته شده و دوباره نصب شدند. داده های ارتعاشی برای بارهای مختلف موتور ثبت شدند. این آزمایش‌ها برای توسعه و اعتبارسنجی تکنیک‌های ارزیابی وضعیت یاتاقان‌ها انجام شد. پایگاه داده‌هایی که از این آزمایش‌ها به دست آمده، برای پروژه‌های مختلف از جمله تکنیک‌های تشخیص مبتنی بر مدل و الگوریتم‌های تعیین سرعت موتور استفاده شده است. داده ها در فرمت Matlab ذخیره شده و شامل اطلاعات سرعت، شتاب و دور در دقیقه هستند.

این پایگاه تست شامل موتور ۲ اسب بخاری، مبدل گشتاور، دینامومتر و کنترل الکترونیکی است. بلبرینگ های موتور محور را پشتیبانی می کنند. خطاهای تک نقطه‌های با قطرهای ۷، ۱۴، ۲۱، ۲۸ و ۴۰ میلی‌متر ایجاد شدند. داده‌های ارتعاشی با شتاب‌سنج‌ها در موقعیت‌های ۱۲ ساعت در انتهای محرک و فن جمع‌آوری و در متلب پردازش شدند. داده‌ها در قالب متلب با نرخ نمونه‌برداری ۱۲۰۰۰ و ۴۸۰۰۰ نمونه بر ذخیره‌سازی شدند. داده های سرعت و اسب بخار با

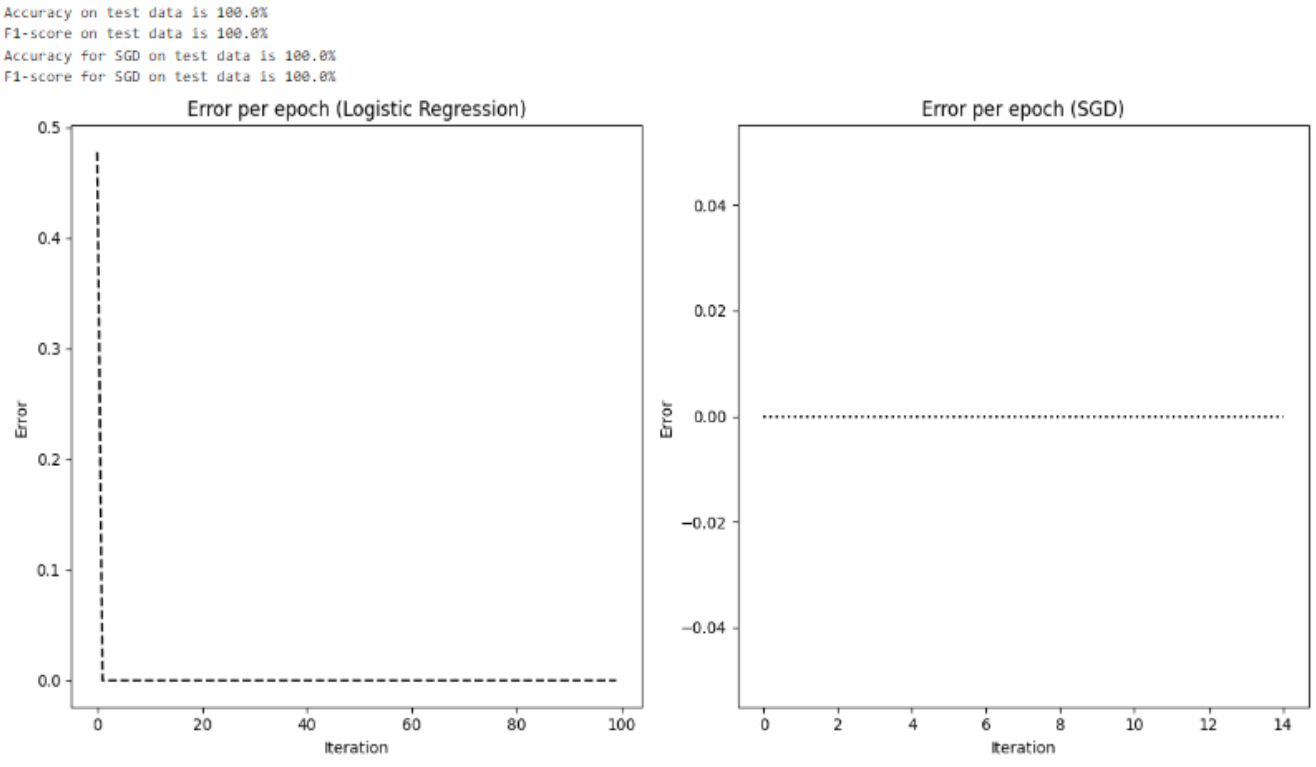
مبدل گشتاور جمع آوری شدند. خطاهای راهرو بیرونی با قرارگیری در ساعت‌های ۳، ۶ و ۱۲ نسبت به ناحیه بار آزمایش می‌شوند تا اثر آسیب خرابی بر پاسخ عاشی بررسی شود.

این مجموعه شامل ۴ کلاس است که هر کلاس به دیتاست هایی با عیوب مختلف تقسیم می شود که میتوانند شامل عیب حلقه داخلی و خارجی، عیب ساچمه و ... باشد.

ابتدا با دستور loadmat فایل ها را میخوانیم و برای استخراج داده، ماتریس ها را تشکیل می دهیم، سپس یکی از کلاس ها ار انتخاب کرده و محاسبات را ادامه می دهیم. برای استخراج ویژگی از توابع از پیش تعریف شده و و فرمول ها استفاده کرده و با عدد گذاری برای آن ها label تعریف کردیم.

در ادامه دیتای مورد نظر را برای پاسخ بهتر شافل کرده و سپس دیتای ترین و ولیدیشن را جدا می کنیم.

سپس دیتاها را برای اینکه ماشین به درستی اعداد و ارزش آن ها را تشخیص دهد، بین -۱ و ۱ نرمالایز می کنیم. در نتیجه کلاس ها تعریف شده و دیتا را روی دو مدل آموزش می دهیم.

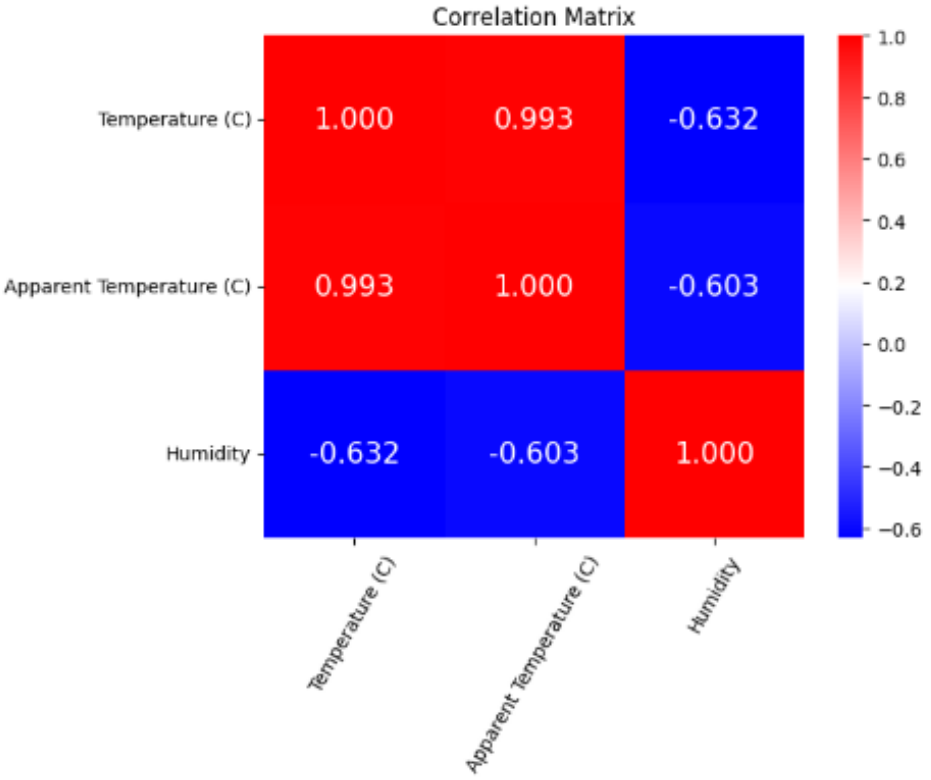


شکل ۱۲: نمودار خطای دو مدل ارائه شده

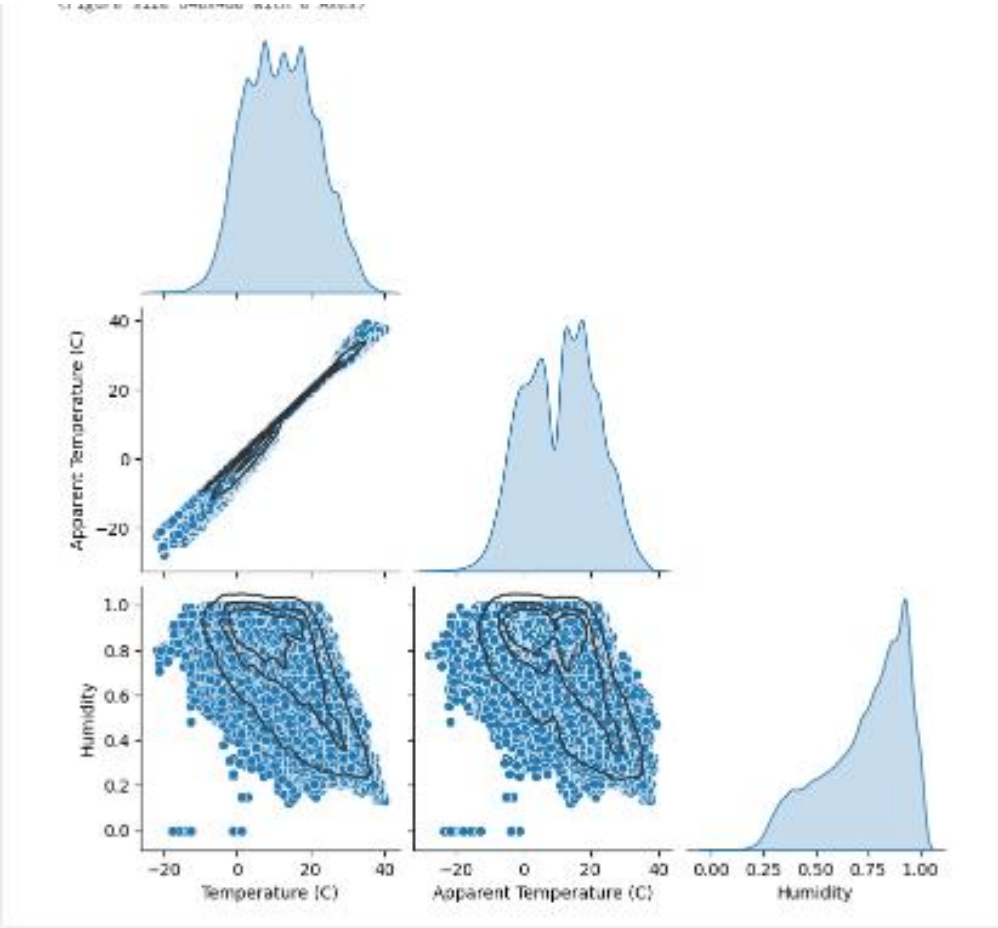
سوال سوم)

دیتا را از مرجع دریافت کرده و با کتابخانه پانداس با استفاده از تابع read-csv فراخوانی میکنیم.

ویژگی های ما ۳ تاست که در شکل۱۳ و۱۴ هیت مپ و هیستوگرام آن رسم شده است.



شکل۱۳: ماتریس همبستگی



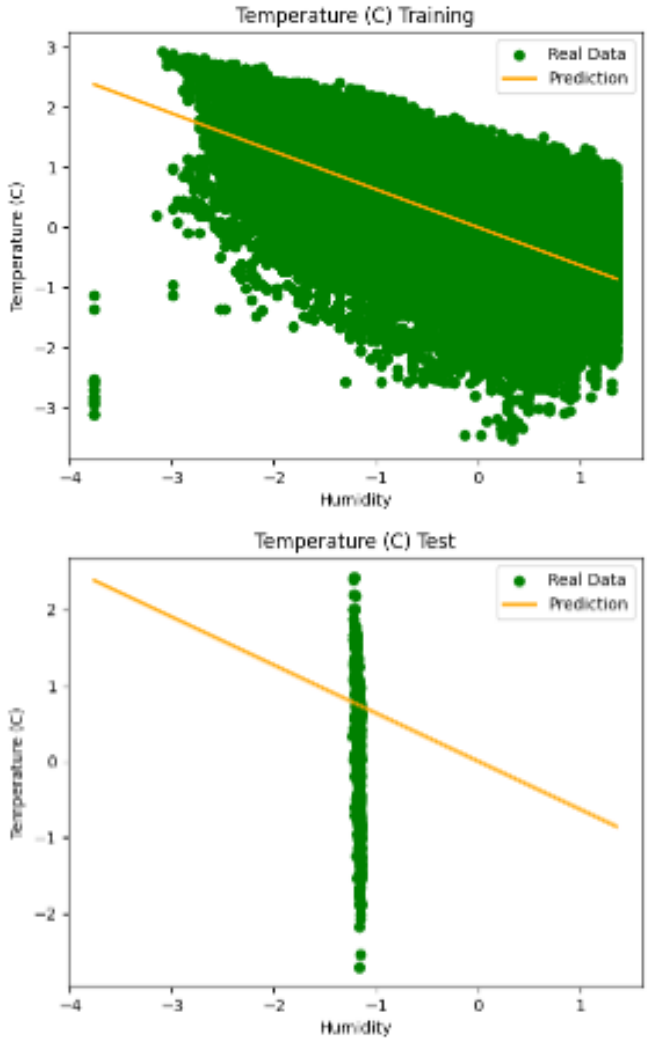
شکل۱۴: پراکندگی دیتا

حال به ترتیب تخمین LS و RLS را روی دیتا اعمال می کنیم.

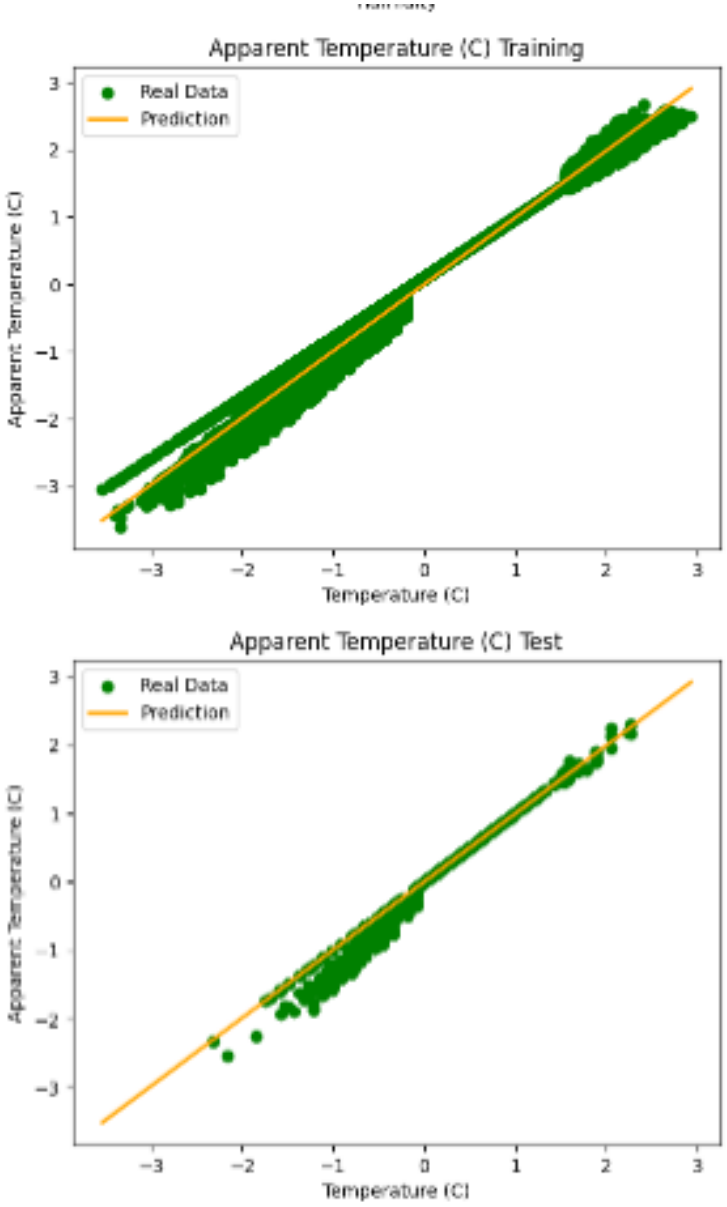
ایجاد کلاس برای LS :

```
class LinearRegressionLS:
    def __init__(self):
        self.coef = None

    def fit(self, x, y):
        x = np.column_stack((np.ones(len(x)), x))
        self.coef = np.linalg.inv(x.T @ x) @ x.T @ y

    def predict(self, x):
        x = np.column_stack((np.ones(len(x)), x))
        return x @ self.coef
```

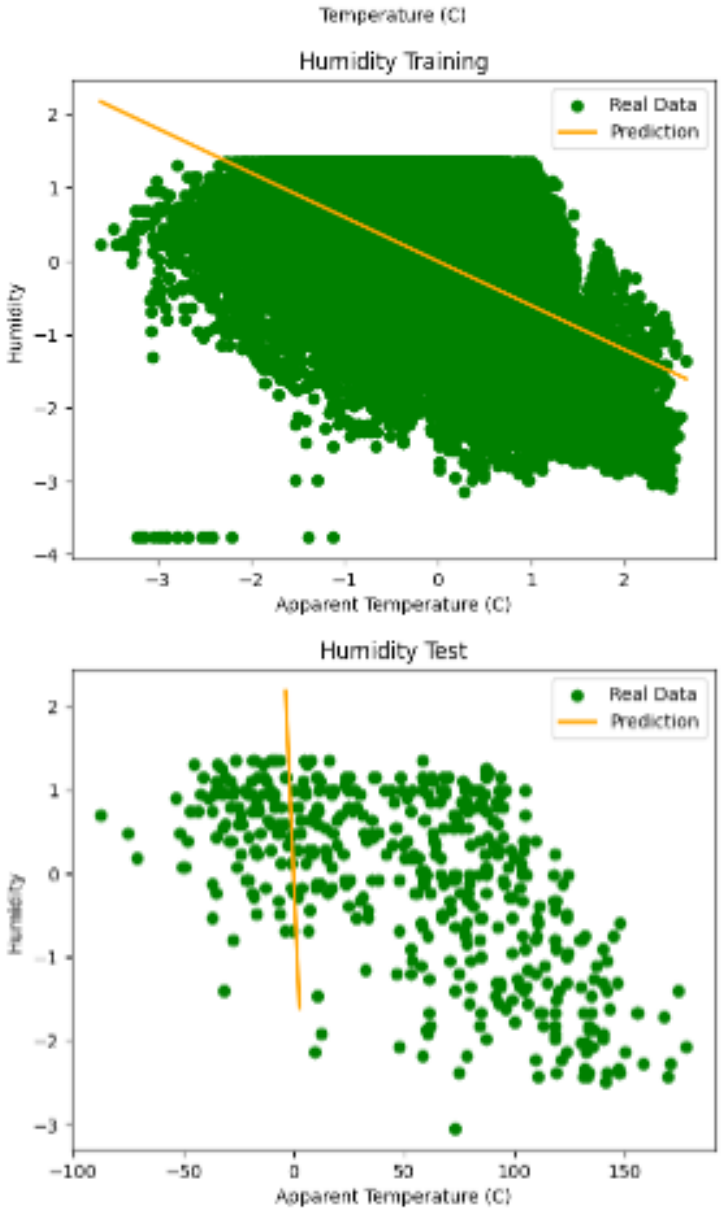
بعد از اسپلیت کردن دیتا و نرمالایز کردن آن ها هر سه ویژگی بر حسب هم در شکل ۱۵ و ۱۶ و ۱۷ رسم شده اند.


Temperature (C) Training
Real Data
Prediction
Temperature (C)
Humidity
Temperature (C) Test
Real Data
Prediction
Temperature (C)
Humidity

شکل۱۵: H&T


Apparent Temperature (C) Training
Real Data
Prediction
Apparent Temperature (C)
Temperature (C)
Apparent Temperature (C) Test
Real Data
Prediction
Apparent Temperature (C)
Temperature (C)

شکل۱۶: A&T



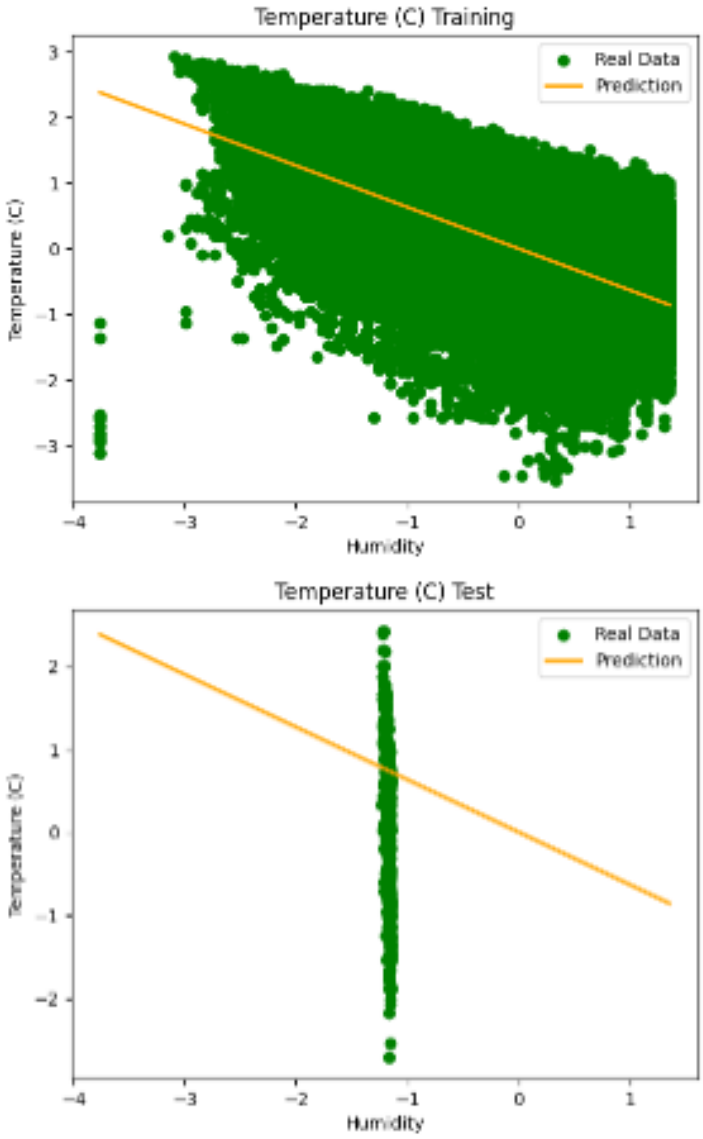
شکل۱۷: H&A


```
{'Temperature (C)': {'MSE Train': 0.6002260407019652, 'MSE Test':
1.5263508963696797, 'MAE Train': 0.6317323768314111, 'MAE Test':
1.0098981493804853}, 'Apparent Temperature (C)': {'MSE Train':
0.014687625389942775, 'MSE Test': 0.03666103172392698, 'MAE Train':
0.0927990097780925, 'MAE Test': 0.11116486483919447}, 'Humidity': {'MSE
Train': 0.6368961953780038, 'MSE Test': 2059.103948792121, 'MAE Train':
0.6469534277977209, 'MAE Test': 37.359025611515726}
```

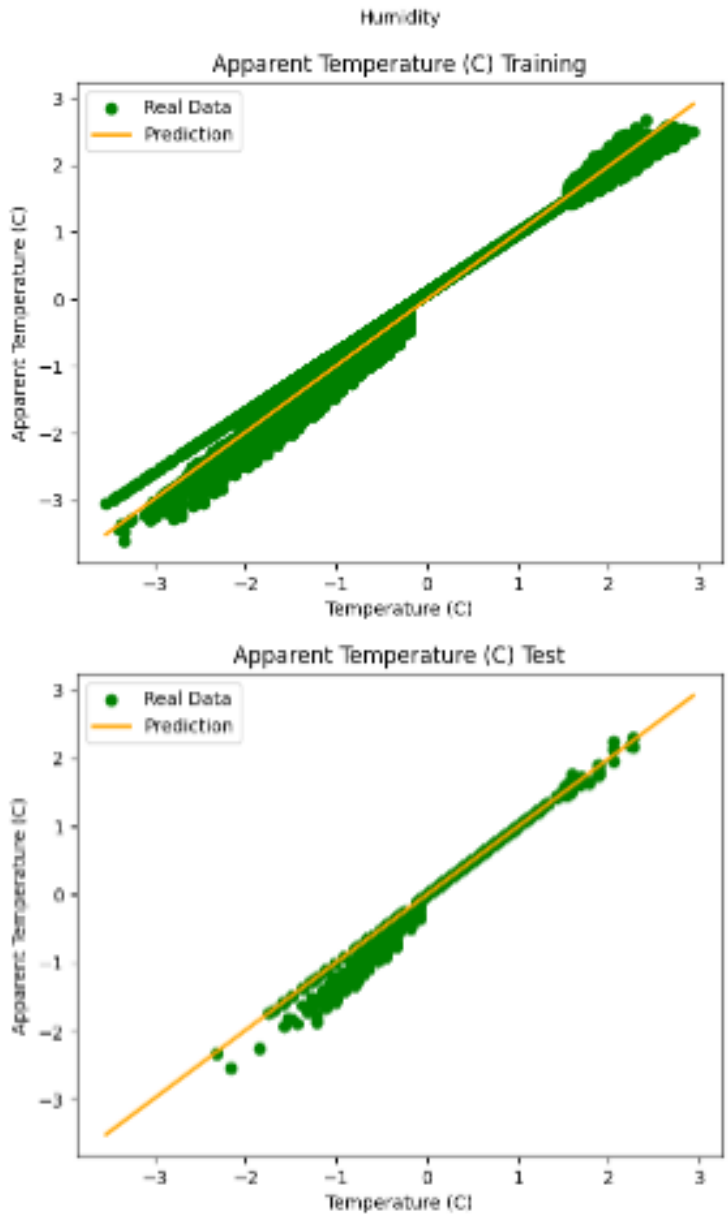

ایجاد کلاس برای RLS:

```python
class LinearRegressionRLS:
    def __init__(self, lambda_=1.0):
        self.lambda_ = lambda_
        self.coef = None

    def fit(self, x, y):
        x = np.column_stack((np.ones(len(x)), x))
        I = np.eye(x.shape[1])
        # Add lambda_ to the diagonal elements of the matrix except the first one for the intercept
        I[0, 0] = 0
        self.coef = np.linalg.inv(x.T @ x + self.lambda_ * I) @ x.T @ y

    def predict(self, x):
        x = np.column_stack((np.ones(len(x)), x))
        return x @ self.coef
```

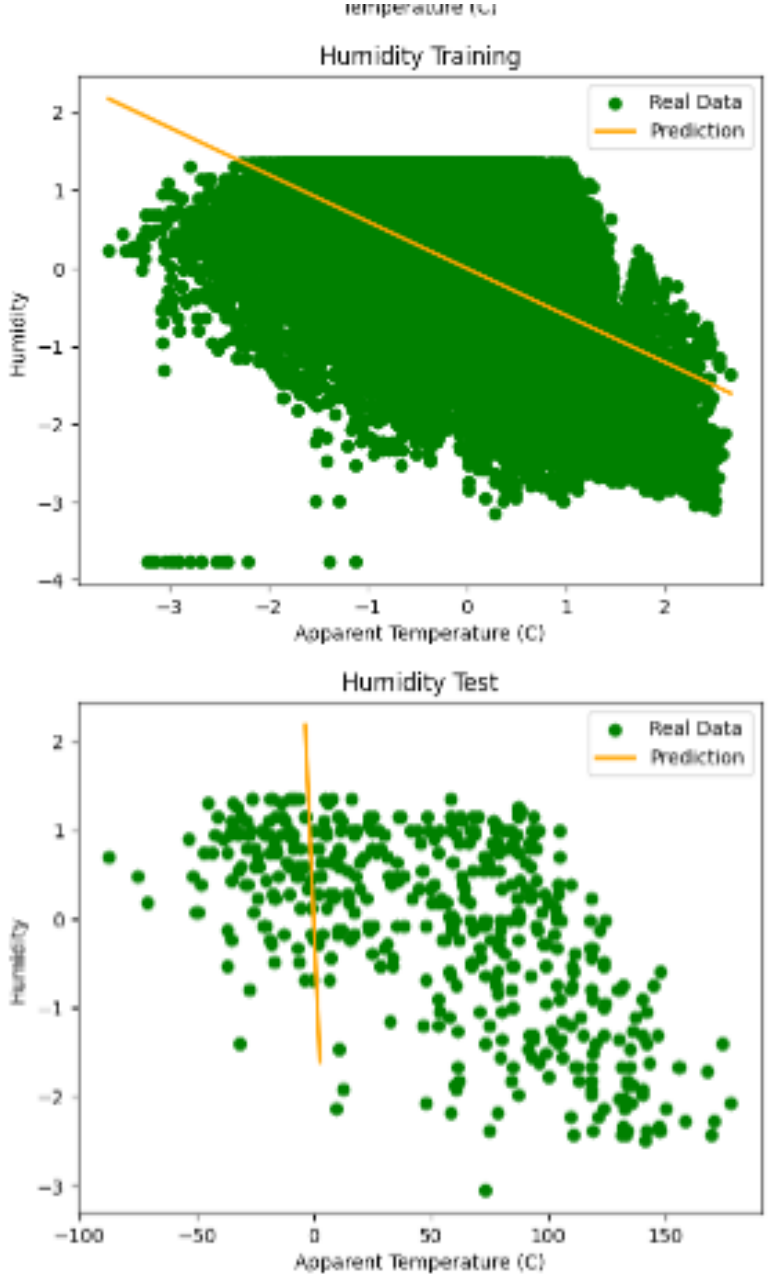
نتایج در شکل ۱۸ و ۱۹ و ۲۰ قابل مشاهده هستند.



شکل۱۸: T&H


Humidity
Apparent Temperature (C) Training
Real Data
Prediction
Apparent Temperature (C)
Temperature (C)
Apparent Temperature (C) Test
Real Data
Prediction
Apparent Temperature (C)
Temperature (C)

شکل۱۹:A&T



شکل ۲۰:A&H


```
{'Temperature (C)': {'MSE Train': 0.6002260407023994, 'MSE Test':
1.5263497926656187, 'MAE Train': 0.6317324193248731, 'MAE Test':
1.0098977681885881}, 'Apparent Temperature (C)': {'MSE Train':
0.01468762539101255, 'MSE Test': 0.03666120384596047, 'MAE Train':
0.09279904505920686, 'MAE Test': 0.1111654783127507 1}, 'Humidity':
{'MSE Train': 0.6368961953783979, 'MSE Test': 2059.0996131577967, 'MAE
Train': 0.6469534576492854, 'MAE Test': 37.35898635443232}
```

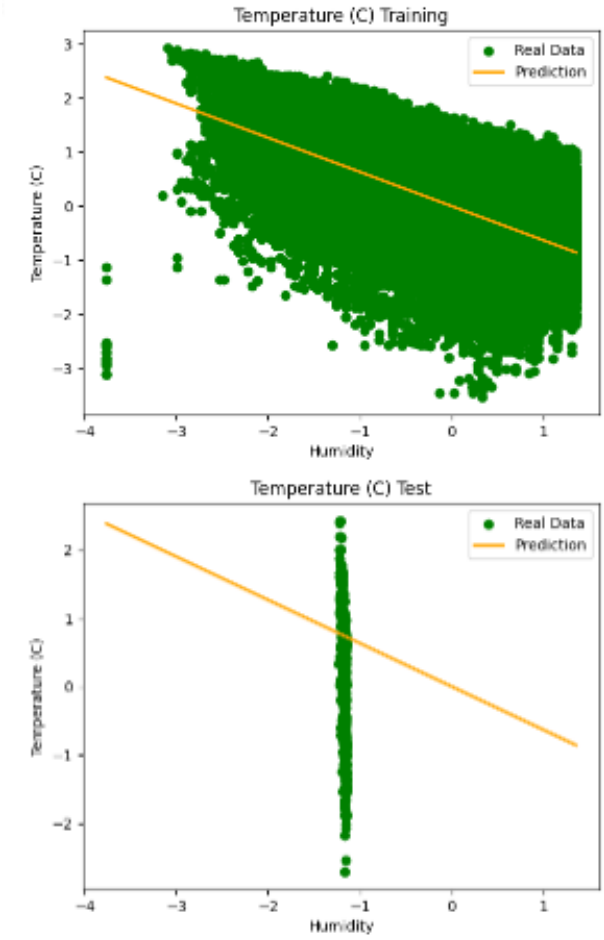

با توجه به شکل های بالا نتیجه می گیریم که چون پراکندگی A و T نسبت به هم کمتر است در نتیجه خطای کمتری نسبت به بقیه دارد.

حال روش WLS را پیاده می کنیم. در این روش فرض می شود که خطا یک واریانس ثابت دارد. کلاس آن را به صورت زیر تعریف می کنیم:

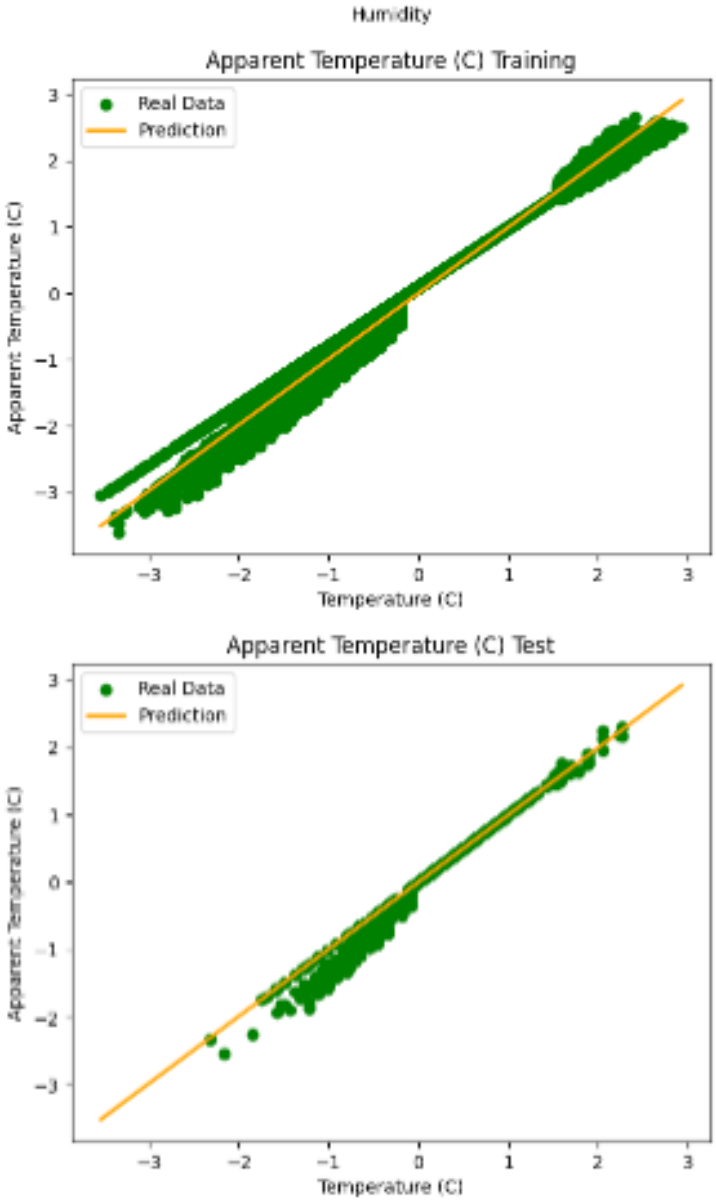
```
class LinearRegressionWLS:
    def __init__(self):
        self.coef = None

    def fit(self, x, y, weights):
        x = np.column_stack((np.ones(len(x)), x))
        W = np.diag(weights.flatten())  # Ensure weights are in diagonal matrix form
        self.coef = np.linalg.inv(x.T @ W @ x) @ x.T @ W @ y

    def predict(self, x):
        x = np.column_stack((np.ones(len(x)), x))
        return x @ self.coef
```

```
# Generate or obtain weights for each data point
weights = np.random.rand(len(input))  # This is a placeholder; use actual logic to set weights
```

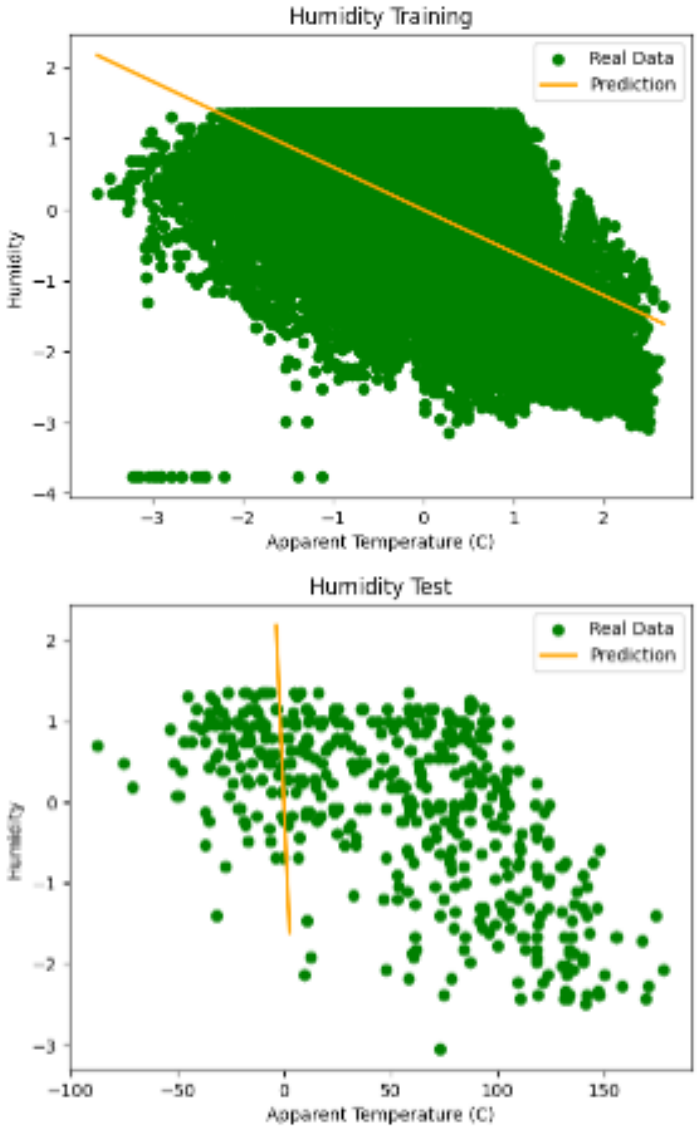
نتایج در شکل ۲۱ و ۲۲ و ۲۳ قابل مشاهده است.



شکل ۲۱: T&H


Humidity
Apparent Temperature (C) Training
Real Data
Prediction
Apparent Temperature (C)
Temperature (C)
Apparent Temperature (C) Test
Real Data
Prediction
Apparent Temperature (C)
Temperature (C)

شکل۲۲: T&A



شکل۲۳:A&H

{'Temperature (C)': {'MSE Train': 0.6002269418030317, 'MSE Test': 1.5257920150379554, 'MAE Train': 0.6317416796937689, 'MAE Test': 1.0097040211709272}, 'Apparent Temperature (C)': {'MSE Train': 0.014687629888120321, 'MSE Test': 0.03664554002592524, 'MAE Train': 0.09279968894424948, 'MAE Test': 0.11112799333418226}, 'Humidity': {'MSE Train': 0.6368994894137605, 'MSE Test': 2062.9428768020616, 'MAE Train': 0.6470277433011012, 'MAE Test': 37.3937492281955}