CSE 4020 Machine Learning
Digital Assignment

Sujay Kumar M 20BDS0294
Computer Science Engineering with Specialization with DataScience
sujaykumarreddy.m2020@vitstudent.ac.in
https://github.com/sujaykumarmag/CSE4020_digital_assignment

April 9, 2023

1 About the Project

1.1 PMV-PPD Prediction

Package to calculate several thermal comfort indices (e.g. PMV, PPD, SET, adaptive) and convert physical variables.

Pythermalcomfort is an open-source Python library that provides a set of tools and functions for assessing thermal comfort in indoor and outdoor environments. The library implements various thermal comfort models, standards, and indices, including ASHRAE-55, ISO 7730, and PMV/PPD. The library is developed and maintained by the Center for the Built Environment (CBE) at the University of California, Berkeley.

The pythermalcomfort library includes the following functionalities:

1. Calculate Thermal Comfort Indices: The library includes functions to calculate various thermal comfort indices, such as Predicted Mean Vote (PMV), Predicted Percentage of Dissatisfied (PPD), and Thermal Sensation Vote (TSV).
2. Estimate Occupant Comfort: The library can estimate the comfort level of occupants based on their thermal sensation votes, clothing insulation, and metabolic rates.
3. Analyze Environmental Parameters: The library includes functions to analyze various environmental parameters, such as air temperature, radiant temperature, air velocity, and humidity.
4. Calculate Energy Consumption: The library can estimate energy consumption for heating and cooling systems based on thermal comfort indices and environmental parameters.
5. Compare Thermal Comfort Models: The library includes functions to compare different thermal comfort models and standards, such as ASHRAE-55 and ISO 7730, and evaluate their performance in different contexts.

1.2 Technical Aspects of this Project

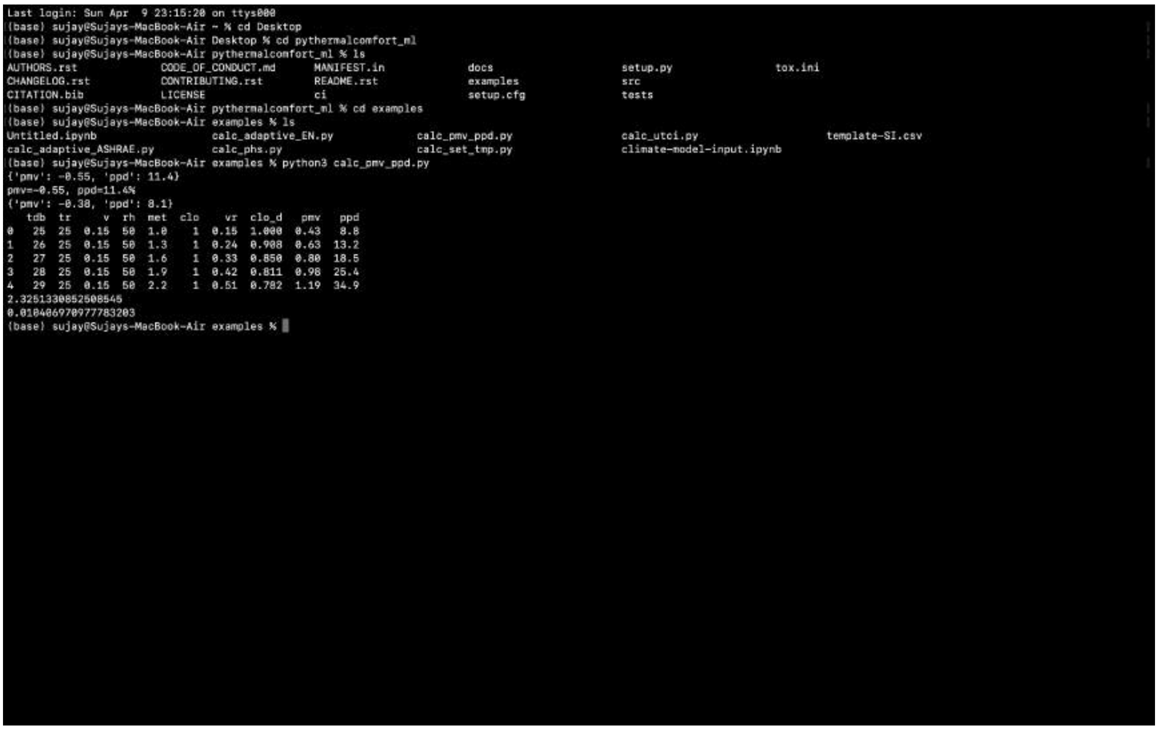
This Project aims to provide a set of tools and functions for evaluating thermal comfort in indoor and outdoor environments, as well as estimating energy consumption for heating and cooling systems.

The library includes various thermal comfort models, such as ASHRAE-55, ISO 7730, and PMV/PPD, which are widely used in the building industry. These models use environmental parameters such as air temperature, radiant temperature, air velocity, and humidity, as well as human factors such as metabolic rate and clothing insulation, to predict the thermal sensation of occupants. The library can estimate the comfort level of occupants based on their thermal sensation votes, clothing insulation, and metabolic rates.

In addition to thermal comfort evaluation, the pythermalcomfort library includes tools for data visualization, such as heatmaps, scatter plots, and histograms. The library can also estimate energy consumption for heating and cooling systems based on thermal comfort indices and environmental parameters.

The library is well-documented, and the API is easy to use, making it accessible to a wide range of users, including architects, engineers, building managers, and researchers. The library is regularly updated with the latest research and standards in the field of thermal comfort, ensuring that it remains relevant and up-to-date.

1.3 Sample Output



```
Last login: Sun Apr  9 23:15:20 on ttys000
(base) sujay@Sujays-MacBook-Air ~ % cd Desktop
(base) sujay@Sujays-MacBook-Air Desktop % cd pythermalcomfort_ml
(base) sujay@Sujays-MacBook-Air pythermalcomfort_ml % ls
AUTHORS.rst          CODE_OF_CONDUCT.md     MANIFEST.in          docs                 setup.py             tox.ini
CHANGELOG.rst        CONTRIBUTING.rst       README.rst           examples             src
CITATION.bib         LICENSE                ci                   setup.cfg            tests
(base) sujay@Sujays-MacBook-Air pythermalcomfort_ml % cd examples
(base) sujay@Sujays-MacBook-Air examples % ls
Untitled.ipynb                 calc_adaptive_EN.py            calc_pmv_ppd.py                calc_utci.py                   template-SI.csv
calc_adaptive_ASHRAE.py        calc_phs.py                    calc_set_tmp.py                climate-model-input.ipynb
(base) sujay@Sujays-MacBook-Air examples % python3 calc_pmv_ppd.py
{'pmv': -0.55, 'ppd': 11.4}
pmv=-0.55, ppd=11.4%
{'pmv': -0.38, 'ppd': 8.1}
   tdb  tr     v  rh  met  clo    vr  clo_d   pmv   ppd
0   25  25  0.15  50  1.0    1  0.15  1.000  0.43   8.8
1   26  25  0.15  50  1.3    1  0.24  0.908  0.63  13.2
2   27  25  0.15  50  1.6    1  0.33  0.850  0.80  18.5
3   28  25  0.15  50  1.9    1  0.42  0.811  0.98  25.4
4   29  25  0.15  50  2.2    1  0.51  0.782  1.19  34.9
2.3251330852508545
0.010406970977783203
(base) sujay@Sujays-MacBook-Air examples %
```

2 Your observation on project and possible additional functionalities

2.1 Functionality 1 - Addition of more Attributes

The addition of more attributes to the calculation of Predicted Mean Vote (PMV) and Predicted Percentage of Dissatisfied (PPD) can enhance the accuracy of thermal comfort evaluations, particularly when using ISO and ASHRAE indices. However, the practical implementation of such enhancements may be constrained by factors such as the availability and cost of sensors and other equipment required to measure and record the additional attributes. Therefore, it is important to carefully consider the benefits and costs of adding more attributes and ensure that the proposed enhancements are feasible and cost-effective for the specific context and requirements of the project.

2.2 Functionality 2 - Machine Learning Aspect

The current version of the repository calculates Predicted Mean Vote (PMV) and Predicted Percentage of Dissatisfied (PPD) and provides this information to the client. However, there is potential to enhance the functionality of the repository by incorporating additional factors that can influence thermal comfort, such as the operation of fans, air conditioning, and windows and doors for airflow, as well as clothing insulation predictions for different types of clothing, and the presence of individuals with medical conditions or of varying ages. By incorporating these factors, the repository can provide a more comprehensive and personalized assessment of thermal comfort that better reflects the needs and preferences of the occupants. However, the incorporation of additional factors may also require additional sensors and data collection equipment, as well as more complex algorithms for analysis and prediction, which may increase the complexity and cost of the system. Therefore, it is important to carefully evaluate the costs and benefits of incorporating additional factors and ensure that any proposed enhancements are feasible, reliable, and effective in meeting the needs of the clients and end-users.

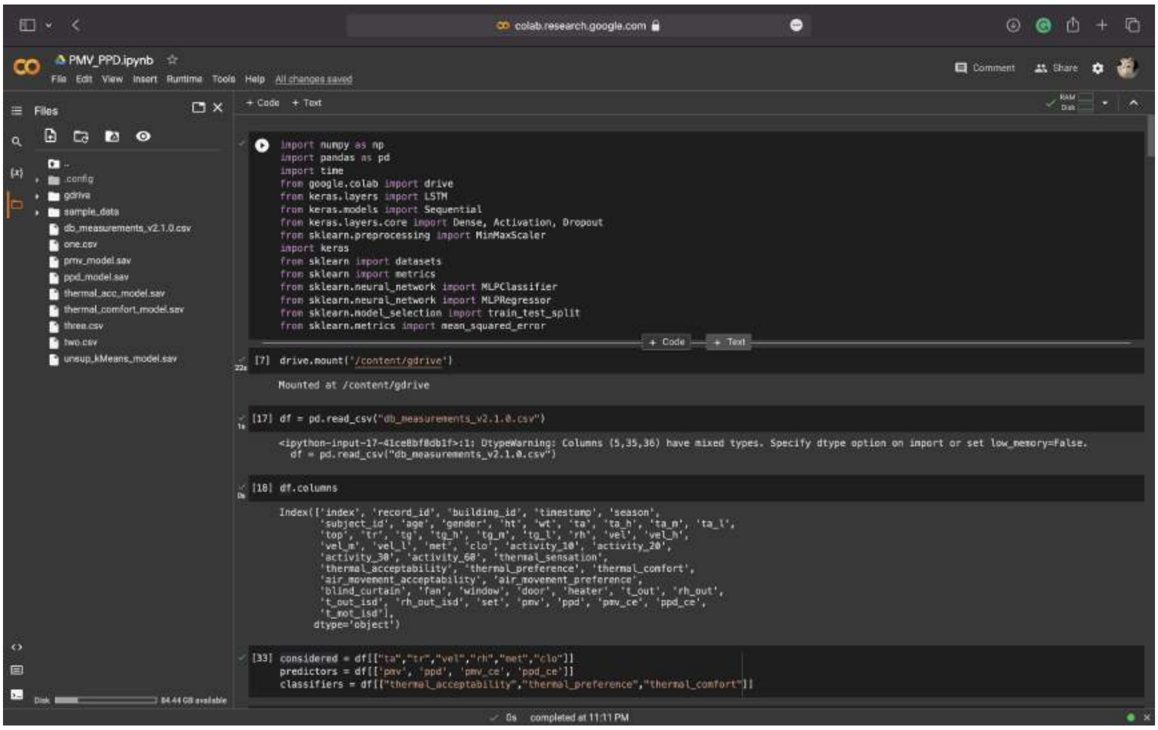
2.3 Functionality 3 - Semi-Supervised Learning

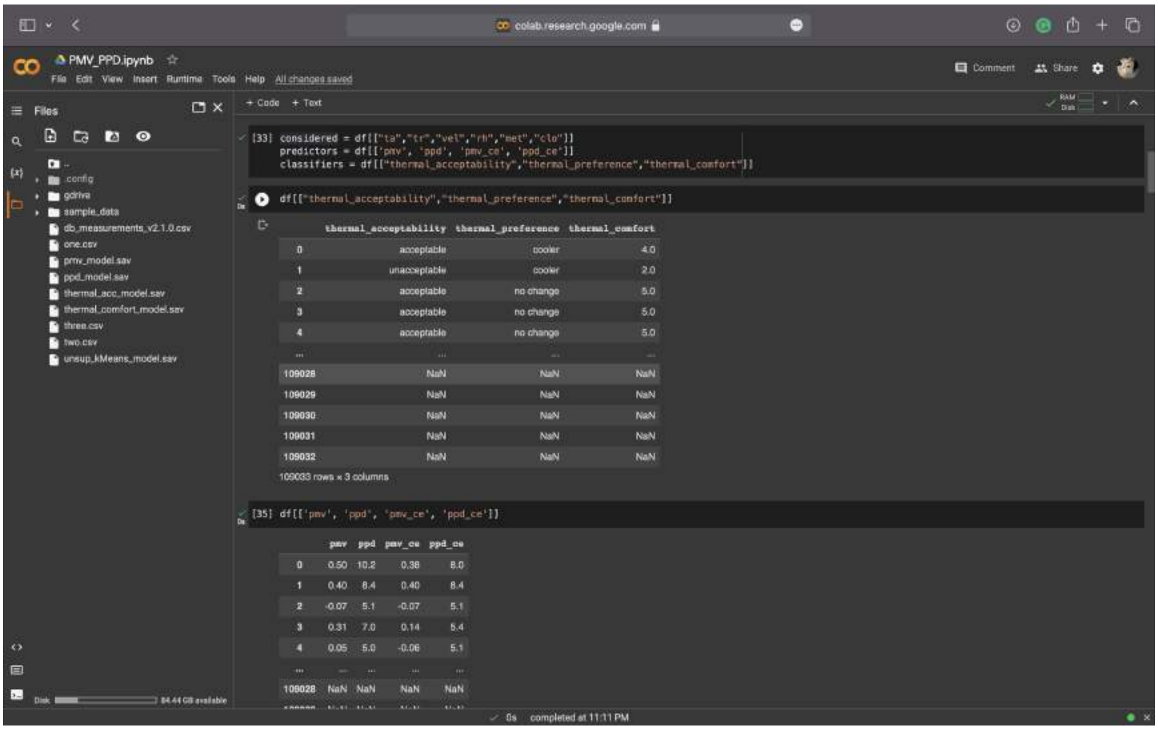
The Usage of Semi-Supervised Learning Comes after the Prediction of PMV and PPD once the Prediction is done we use the Semi-Supervised Learning to Classify Is the Thermal sensation is good-enough or not or if its an yes/no then on which level and by this we can conclude this DataStream Application from the Sensors give us an enumerate approach to predict and classify for the persons living inside the room.

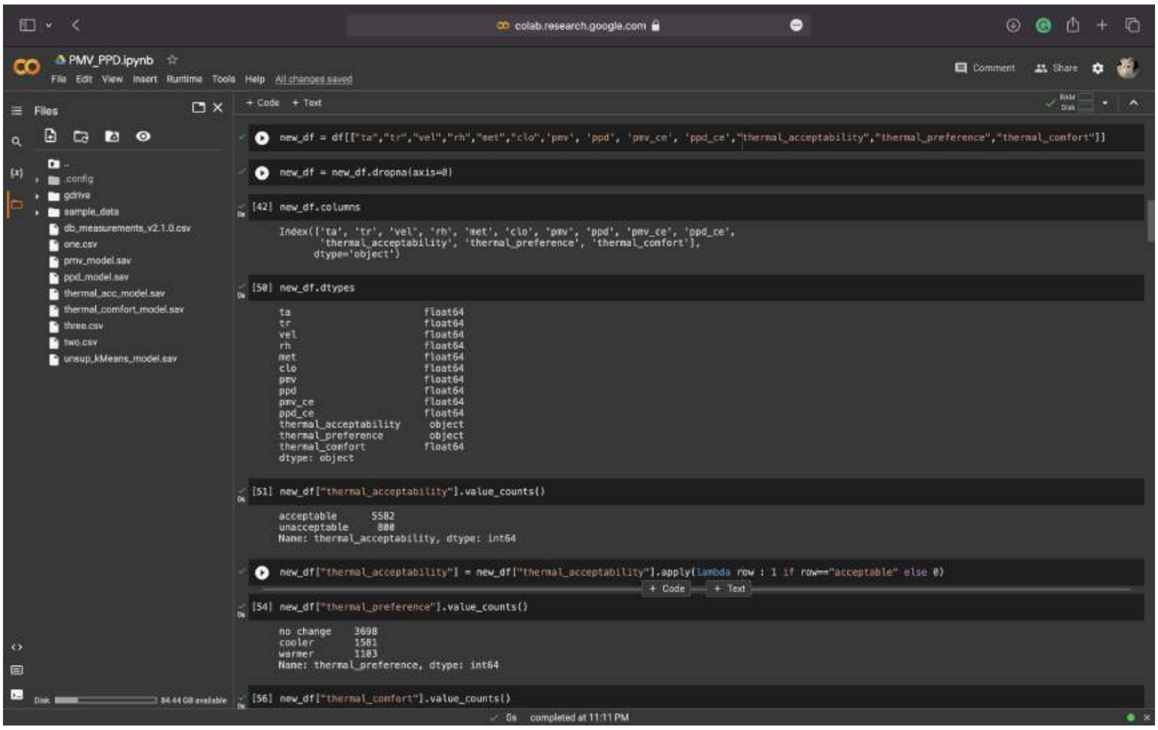
The application of semi-supervised learning can be a valuable addition to the existing prediction of Predicted Mean Vote (PMV) and Predicted Percentage of Dissatisfied (PPD) in thermal comfort assessment. After the prediction of PMV and PPD, semi-supervised learning can be used to classify the thermal sensation as good enough or not and, if not, at which level. This can provide a more detailed and nuanced evaluation of thermal comfort that is tailored to the specific needs and preferences of the occupants. The incorporation of semi-supervised learning can also allow for a more dynamic and responsive system that can adapt to changes in the environment and the occupants over time.

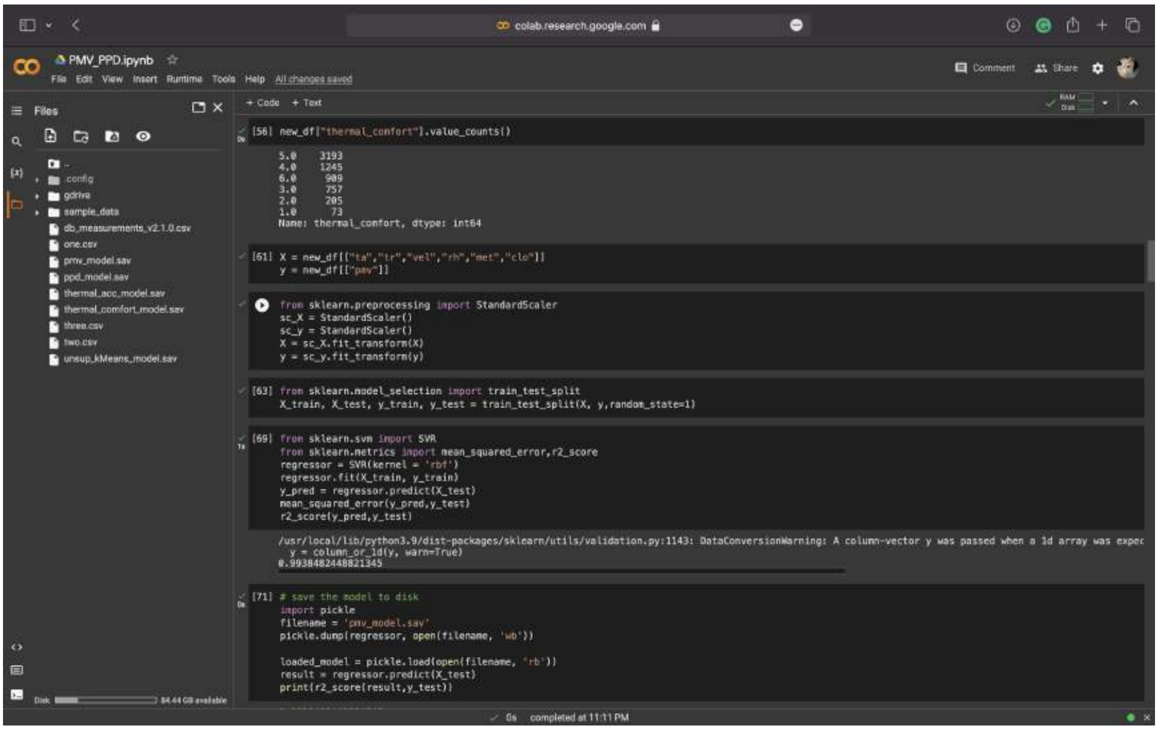
By leveraging data from sensors in real-time, the data stream application can provide a more accurate and up-to-date assessment of thermal comfort, which is particularly valuable in dynamic and unpredictable environments. However, the implementation of semi-supervised learning may require additional computational resources and expertise in machine learning, which can increase the complexity and cost of the system. Therefore, it is important to carefully consider the costs and benefits of incorporating semi-supervised learning and ensure that the proposed enhancements are feasible, reliable, and effective in meeting the needs of the clients and end-users.

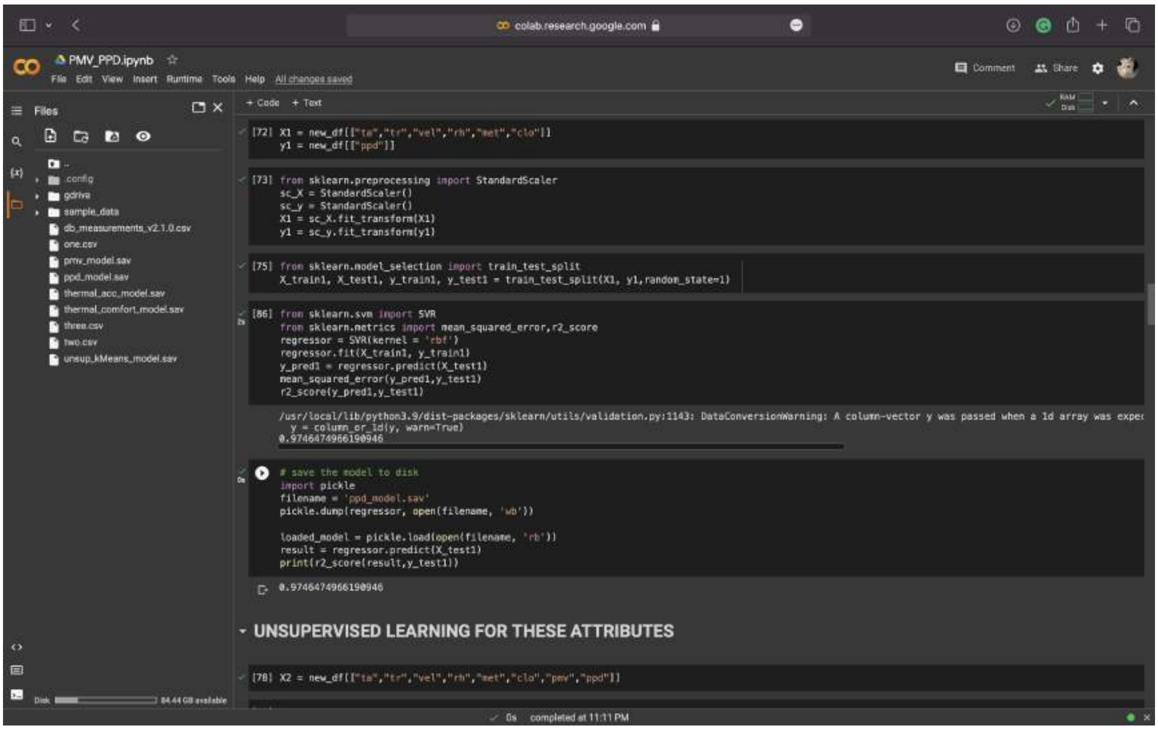
3 My Feature

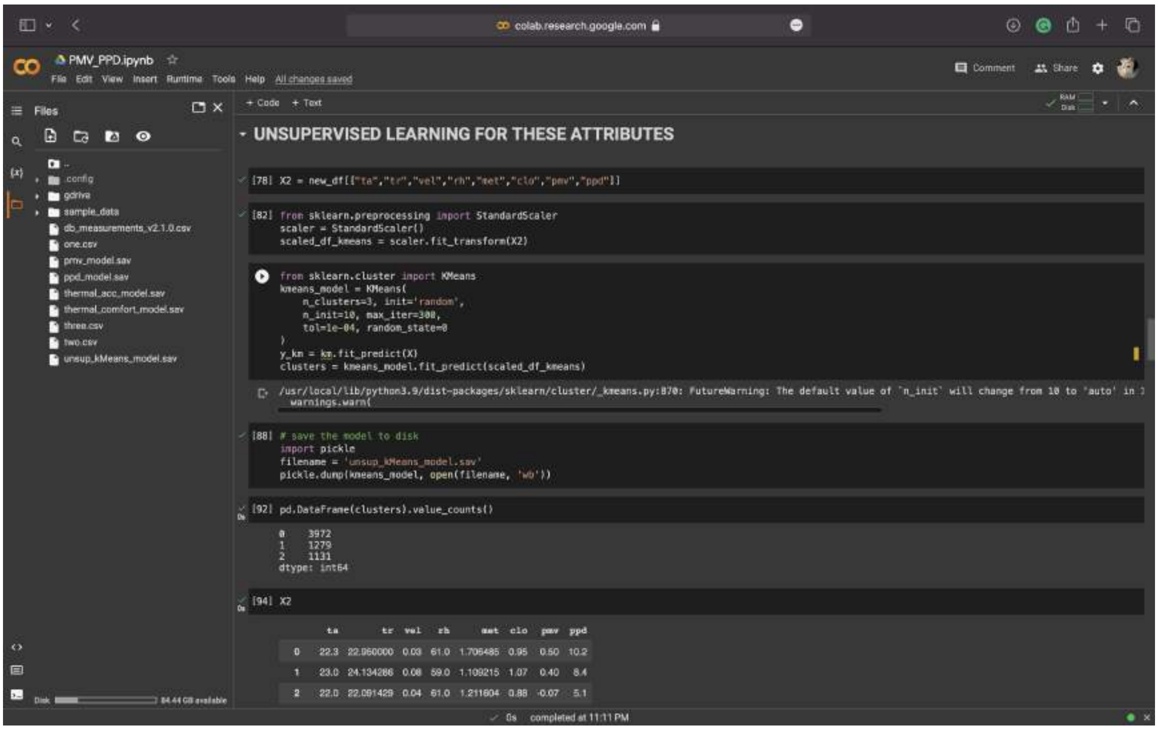

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
import numpy as np
import pandas as pd
import time
from google.colab import drive
from keras.layers import LSTM
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Dropout
from sklearn.preprocessing import MinMaxScaler
import keras
from sklearn import datasets
from sklearn import metrics
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
[7] drive.mount('/content/gdrive')
Mounted at /content/gdrive
[17] df = pd.read_csv("db_measurements_v2.1.0.csv")
<ipython-input-17-41ce8bf8db1f>:1: DtypeWarning: Columns (5,35,36) have mixed types. Specify dtype option on import or set low_memory=False.
df = pd.read_csv("db_measurements_v2.1.0.csv")
[18] df.columns
Index(['index', 'record_id', 'building_id', 'timestamp', 'season', 'subject_id', 'age', 'gender', 'ht', 'wt', 'ta', 'ta_h', 'ta_m', 'ta_l', 'top', 'tr', 'tg', 'tg_h', 'tg_m', 'tg_l', 'rh', 'vel', 'vel_h', 'vel_m', 'vel_l', 'met', 'clo', 'activity_10', 'activity_20', 'activity_30', 'activity_60', 'thermal_sensation', 'thermal_acceptability', 'thermal_preference', 'thermal_comfort', 'air_movement_acceptability', 'air_movement_preference', 'blind_curtain', 'fan', 'window', 'door', 'heater', 't_out', 'rh_out', 't_out_isd', 'rh_out_isd', 'set', 'pmv', 'ppd', 'pmv_ce', 'ppd_ce', 't_mot_isd'],
dtype='object')
[33] considered = df[["ta","tr","vel","rh","met","clo"]]
predictors = df[['pmv', 'ppd', 'pmv_ce', 'ppd_ce']]
classifiers = df[["thermal_acceptability","thermal_preference","thermal_comfort"]]
completed at 11:11 PM

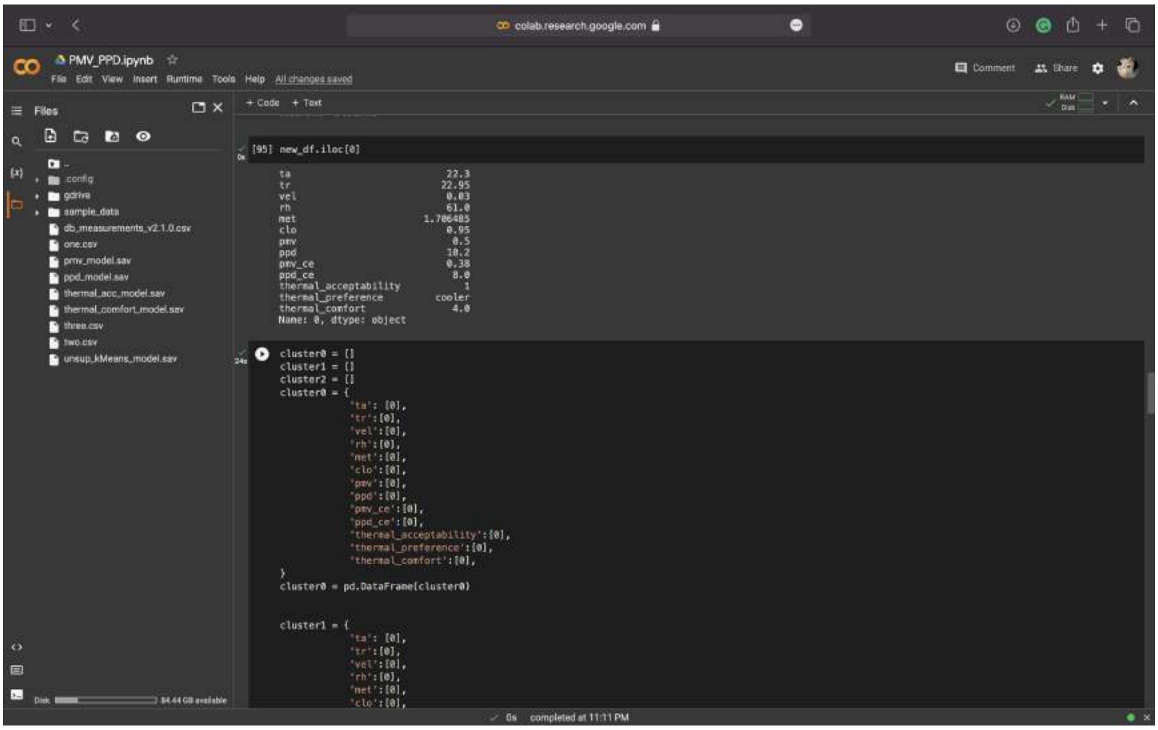

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
[33] considered = df[["ta","tr","vel","rh","met","clo"]]
predictors = df[['pmv', 'ppd', 'pmv_ce', 'ppd_ce']]
classifiers = df[["thermal_acceptability","thermal_preference","thermal_comfort"]]
df[["thermal_acceptability","thermal_preference","thermal_comfort"]]
thermal_acceptability thermal_preference thermal_comfort
0 acceptable cooler 4.0
1 unacceptable cooler 2.0
2 acceptable no change 5.0
3 acceptable no change 5.0
4 acceptable no change 5.0
109028 NaN NaN NaN
109029 NaN NaN NaN
109030 NaN NaN NaN
109031 NaN NaN NaN
109032 NaN NaN NaN
109033 rows × 3 columns
[35] df[['pmv', 'ppd', 'pmv_ce', 'ppd_ce']]
pmv ppd pmv_ce ppd_ce
0 0.50 10.2 0.38 8.0
1 0.40 8.4 0.40 8.4
2 -0.07 5.1 -0.07 5.1
3 0.31 7.0 0.14 5.4
4 0.05 5.0 -0.06 5.1
109028 NaN NaN NaN NaN
completed at 11:11 PM

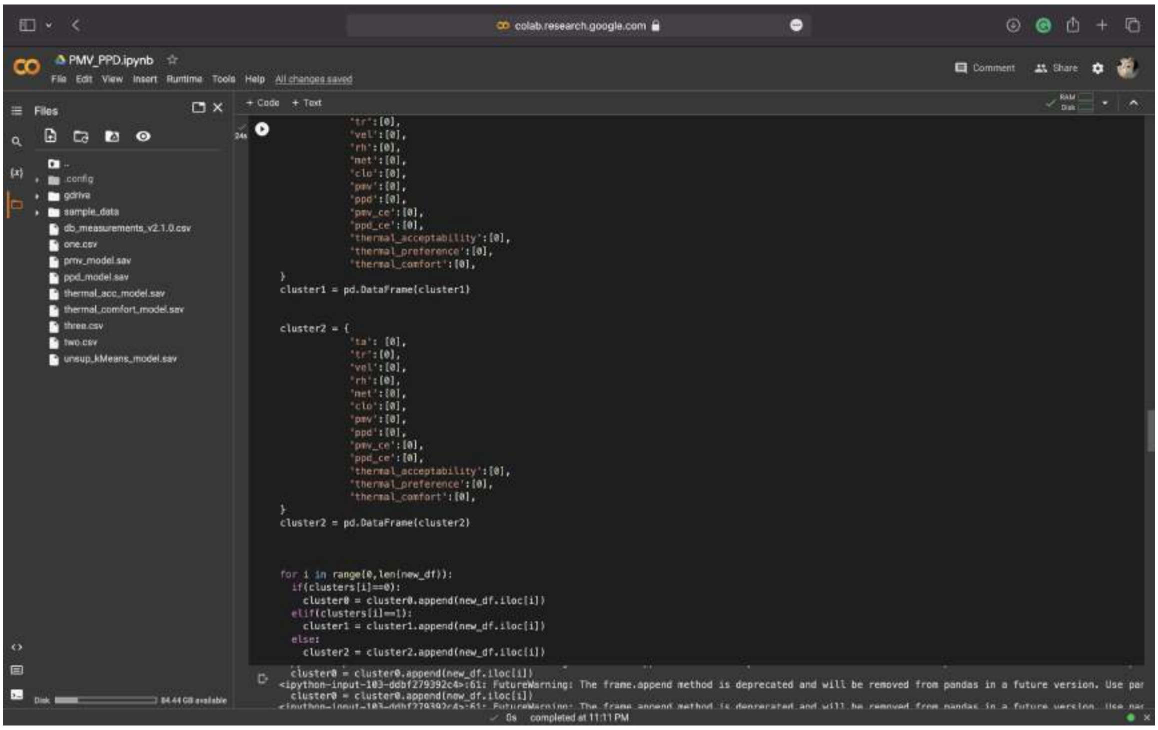

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
new_df = df[["ta","tr","vel","rh","met","clo",'pmv', 'ppd', 'pmv_ce', 'ppd_ce',"thermal_acceptability","thermal_preference","thermal_comfort"]]
new_df = new_df.dropna(axis=0)
[42] new_df.columns
Index(['ta', 'tr', 'vel', 'rh', 'met', 'clo', 'pmv', 'ppd', 'pmv_ce', 'ppd_ce', 'thermal_acceptability', 'thermal_preference', 'thermal_comfort'], dtype='object')
[50] new_df.dtypes
ta float64
tr float64
vel float64
rh float64
met float64
clo float64
pmv float64
ppd float64
pmv_ce float64
ppd_ce float64
thermal_acceptability object
thermal_preference object
thermal_comfort float64
dtype: object
[51] new_df["thermal_acceptability"].value_counts()
acceptable 5582
unacceptable 800
Name: thermal_acceptability, dtype: int64
new_df["thermal_acceptability"] = new_df["thermal_acceptability"].apply(lambda row : 1 if row=="acceptable" else 0)
[54] new_df["thermal_preference"].value_counts()
no change 3698
cooler 1581
warmer 1103
Name: thermal_preference, dtype: int64
[56] new_df["thermal_comfort"].value_counts()
0s completed at 11:11 PM

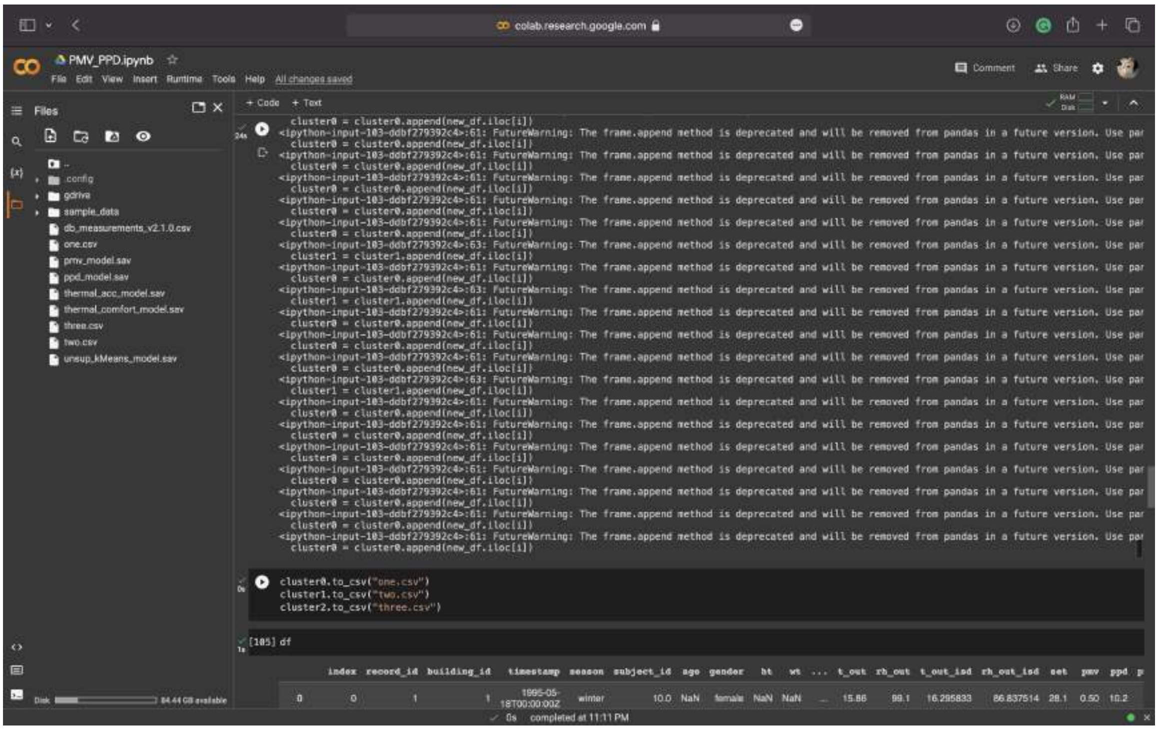

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
[56] new_df["thermal_comfort"].value_counts()
5.0 3193
4.0 1245
6.0 989
3.0 757
2.0 205
1.0 73
Name: thermal_comfort, dtype: int64
[61] X = new_df[["ta","tr","vel","rh","met","clo"]]
y = new_df[["pmv"]]
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
[63] from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=1)
[69] from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error,r2_score
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
mean_squared_error(y_pred,y_test)
r2_score(y_pred,y_test)
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expec
y = column_or_1d(y, warn=True)
0.9938482448821345
[71] # save the model to disk
import pickle
filename = 'pmv_model.sav'
pickle.dump(regressor, open(filename, 'wb'))
loaded_model = pickle.load(open(filename, 'rb'))
result = regressor.predict(X_test)
print(r2_score(result,y_test))
0s completed at 11:11 PM

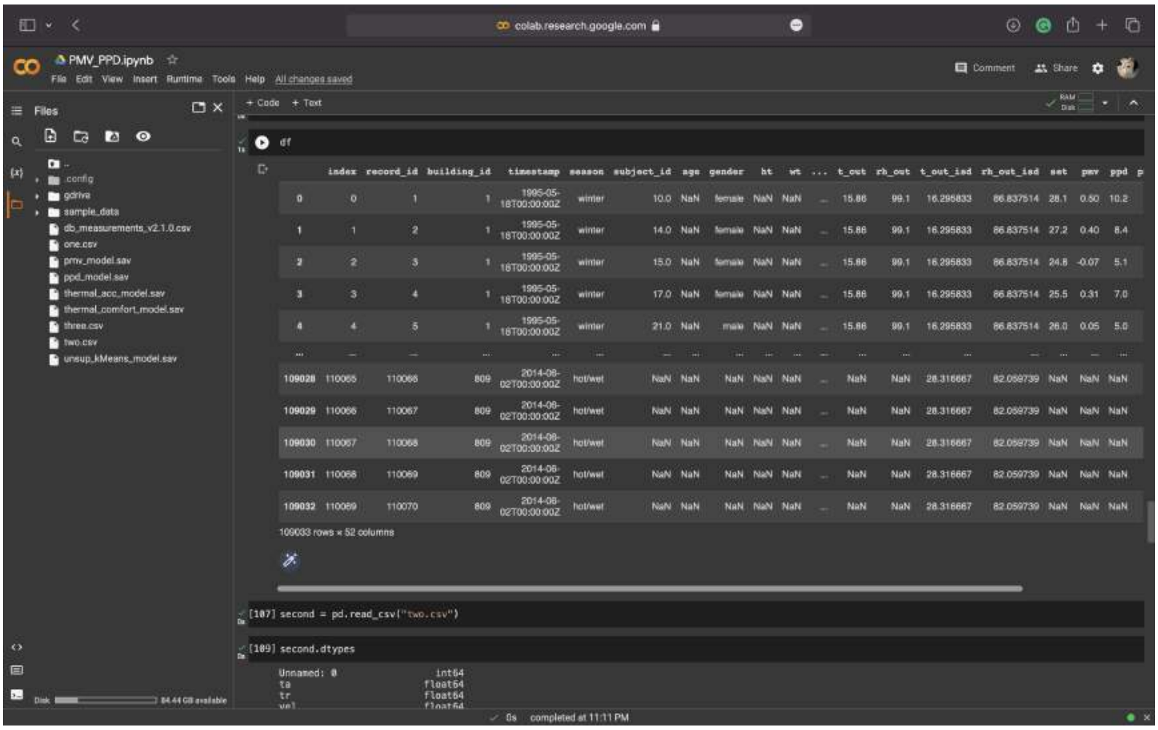

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
config
gdrive
sample_data
db_measurements_v2.1.0.csv
one.csv
pmv_model.sav
ppd_model.sav
thermal_acc_model.sav
thermal_comfort_model.sav
three.csv
two.csv
unsup_kMeans_model.sav
+ Code + Text
[72] X1 = new_df[["ta","tr","vel","rh","met","clo"]]
y1 = new_df[["ppd"]]
[73] from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X1 = sc_X.fit_transform(X1)
y1 = sc_y.fit_transform(y1)
[75] from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1,random_state=1)
[86] from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error,r2_score
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train1, y_train1)
y_pred1 = regressor.predict(X_test1)
mean_squared_error(y_pred1,y_test1)
r2_score(y_pred1,y_test1)
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expe
y = column_or_1d(y, warn=True)
0.9746474966190946
# save the model to disk
import pickle
filename = 'ppd_model.sav'
pickle.dump(regressor, open(filename, 'wb'))
loaded_model = pickle.load(open(filename, 'rb'))
result = regressor.predict(X_test1)
print(r2_score(result,y_test1))
0.9746474966190946
UNSUPERVISED LEARNING FOR THESE ATTRIBUTES
[78] X2 = new_df[["ta","tr","vel","rh","met","clo","pmv","ppd"]]
completed at 11:11 PM

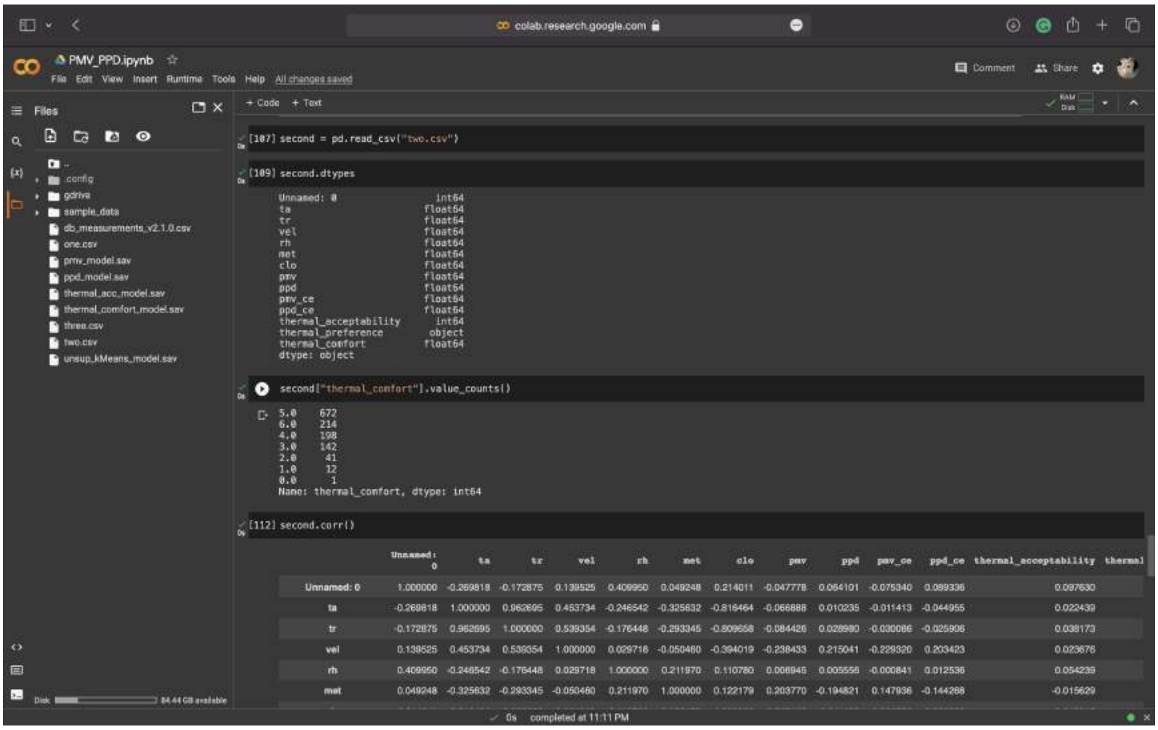

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
config
gdrive
sample_data
db_measurements_v2.1.0.csv
one.csv
pmv_model.sav
ppd_model.sav
thermal_acc_model.sav
thermal_comfort_model.sav
three.csv
two.csv
unsup_kMeans_model.sav
+ Code + Text
UNSUPERVISED LEARNING FOR THESE ATTRIBUTES
[78] X2 = new_df[["ta","tr","vel","rh","met","clo","pmv","ppd"]]
[82] from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_df_kmeans = scaler.fit_transform(X2)
from sklearn.cluster import KMeans
kmeans_model = KMeans(
n_clusters=3, init='random',
n_init=10, max_iter=300,
tol=1e-04, random_state=0
)
y_km = km.fit_predict(X)
clusters = kmeans_model.fit_predict(scaled_df_kmeans)
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
warnings.warn(
[88] # save the model to disk
import pickle
filename = 'unsup_kMeans_model.sav'
pickle.dump(kmeans_model, open(filename, 'wb'))
[92] pd.DataFrame(clusters).value_counts()
0 3972
1 1279
2 1131
dtype: int64
[94] X2
ta tr vel rh met clo pmv ppd
0 22.3 22.960000 0.03 61.0 1.706485 0.95 0.50 10.2
1 23.0 24.134286 0.08 59.0 1.109215 1.07 0.40 8.4
2 22.0 22.091429 0.04 61.0 1.211604 0.88 -0.07 5.1
completed at 11:11 PM

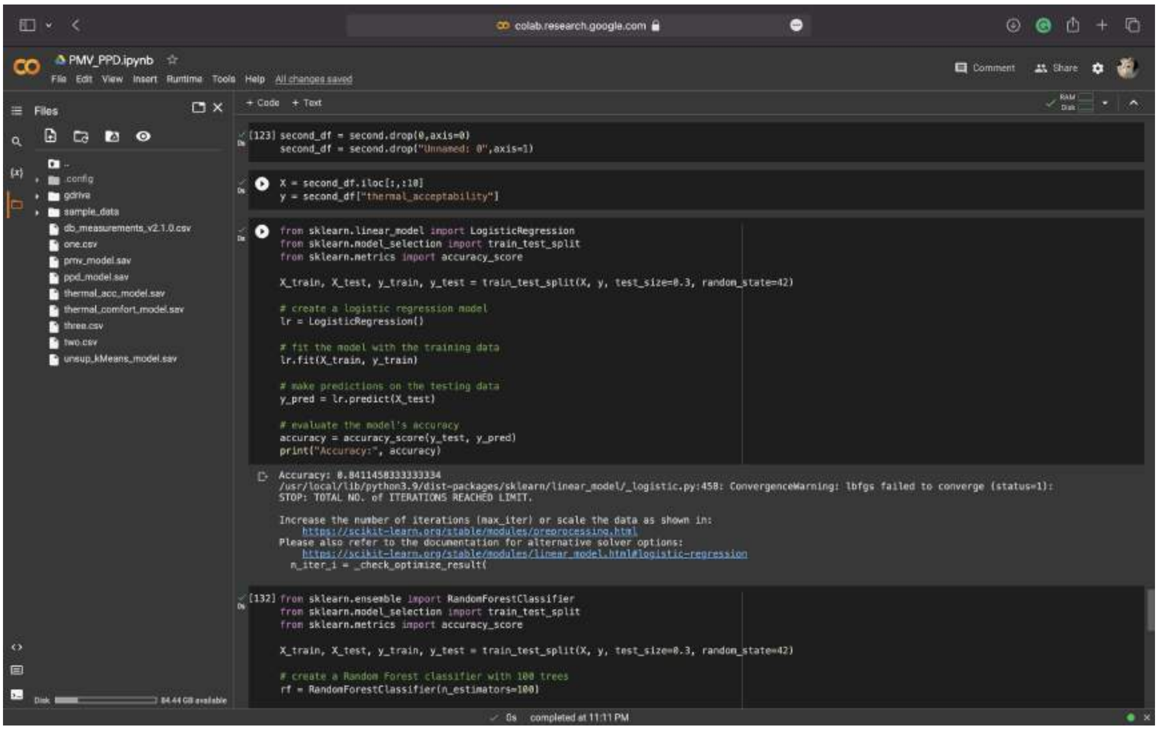

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
config
gdrive
sample_data
db_measurements_v2.1.0.csv
one.csv
pmv_model.sav
ppd_model.sav
thermal_acc_model.sav
thermal_comfort_model.sav
three.csv
two.csv
unsup_kMeans_model.sav
[95] new_df.iloc[0]
ta 22.3
tr 22.95
vel 0.03
rh 61.0
met 1.706485
clo 0.95
pmv 0.5
ppd 10.2
pmv_ce 0.38
ppd_ce 8.0
thermal_acceptability 1
thermal_preference cooler
thermal_comfort 4.0
Name: 0, dtype: object
cluster0 = []
cluster1 = []
cluster2 = []
cluster0 = {
'ta': [0],
'tr':[0],
'vel':[0],
'rh':[0],
'met':[0],
'clo':[0],
'pmv':[0],
'ppd':[0],
'pmv_ce':[0],
'ppd_ce':[0],
'thermal_acceptability':[0],
'thermal_preference':[0],
'thermal_comfort':[0],
}
cluster0 = pd.DataFrame(cluster0)
cluster1 = {
'ta': [0],
'tr':[0],
'vel':[0],
'rh':[0],
'met':[0],
'clo':[0],
completed at 11:11 PM

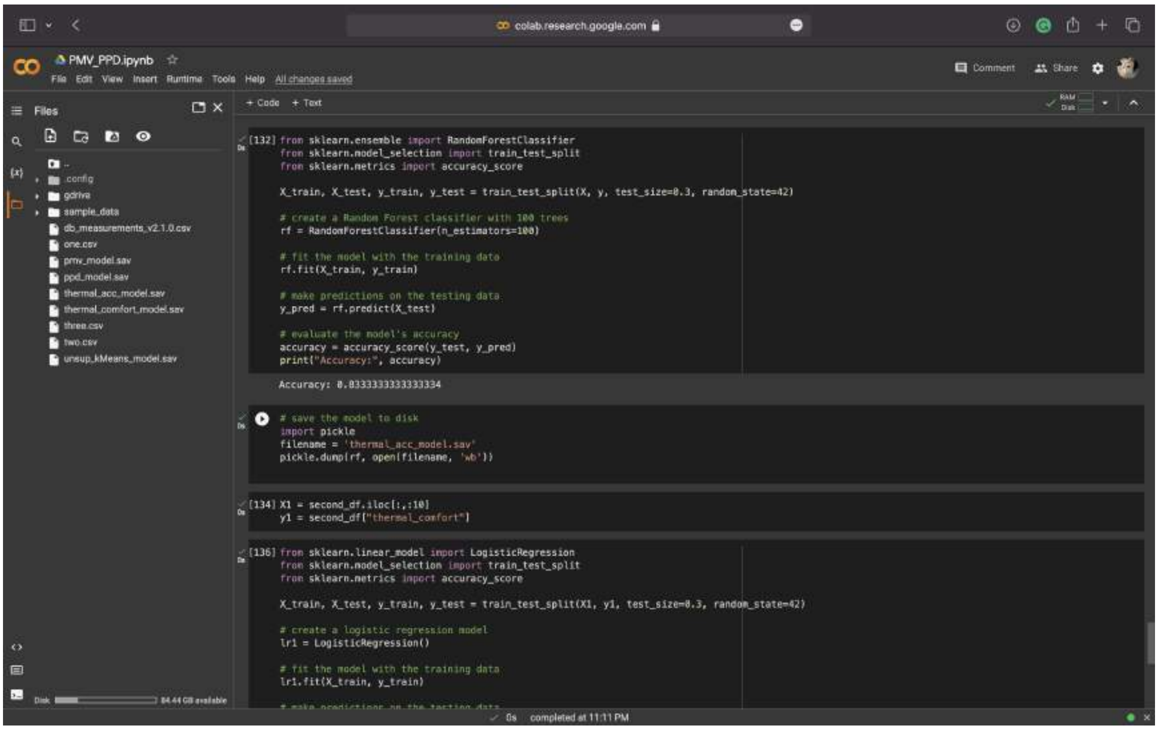

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
'tr':[0],
'vel':[0],
'rh':[0],
'met':[0],
'clo':[0],
'pmv':[0],
'ppd':[0],
'pmv_ce':[0],
'ppd_ce':[0],
'thermal_acceptability':[0],
'thermal_preference':[0],
'thermal_comfort':[0],
}
cluster1 = pd.DataFrame(cluster1)
cluster2 = {
'ta': [0],
'tr':[0],
'vel':[0],
'rh':[0],
'met':[0],
'clo':[0],
'pmv':[0],
'ppd':[0],
'pmv_ce':[0],
'ppd_ce':[0],
'thermal_acceptability':[0],
'thermal_preference':[0],
'thermal_comfort':[0],
}
cluster2 = pd.DataFrame(cluster2)
for i in range(0,len(new_df)):
if(clusters[i]==0):
cluster0 = cluster0.append(new_df.iloc[i])
elif(clusters[i]==1):
cluster1 = cluster1.append(new_df.iloc[i])
else:
cluster2 = cluster2.append(new_df.iloc[i])
cluster0 = cluster0.append(new_df.iloc[i])
<ipython-input-103-ddbf279392c4>:61: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pan
completed at 11:11 PM

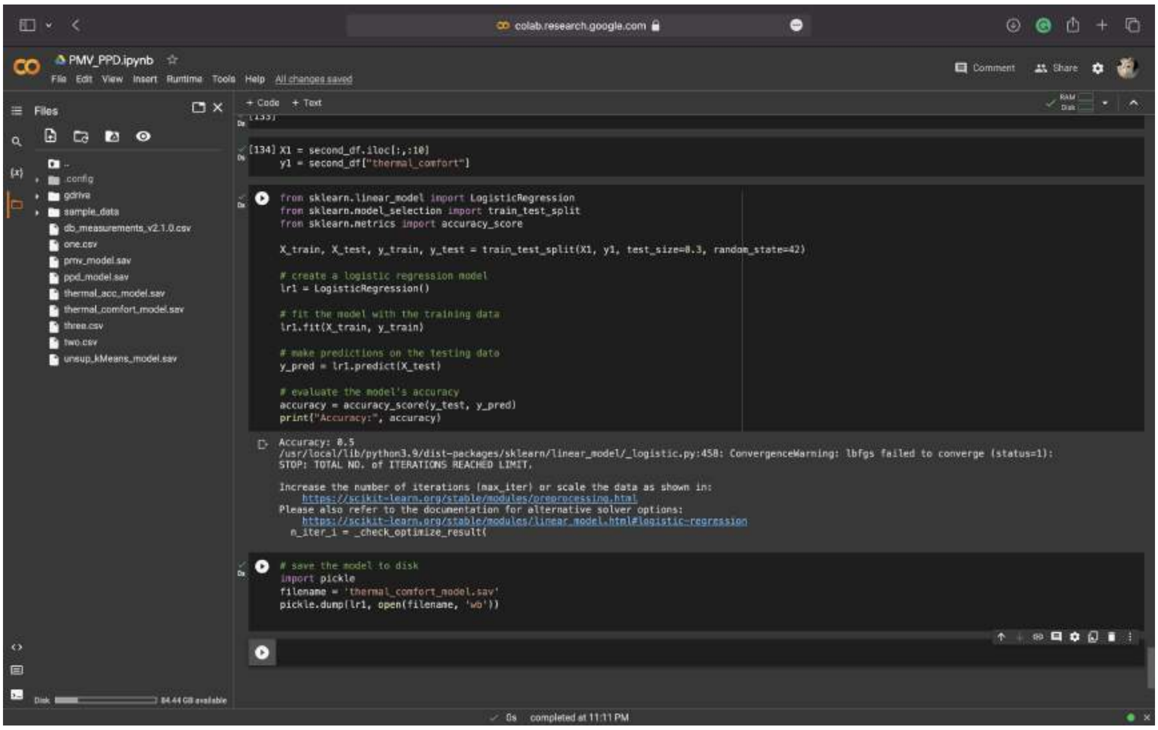

colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
cluster0.to_csv("one.csv")
cluster1.to_csv("two.csv")
cluster2.to_csv("three.csv")
[105] df
completed at 11:11 PM


colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
df
109033 rows × 52 columns
[107] second = pd.read_csv("two.csv")
[109] second.dtypes
completed at 11:11 PM


colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
.config
gdrive
sample_data
db_measurements_v2.1.0.csv
one.csv
pmv_model.sav
ppd_model.sav
thermal_acc_model.sav
thermal_comfort_model.sav
three.csv
two.csv
unsup_kMeans_model.sav
[107] second = pd.read_csv("two.csv")
[109] second.dtypes
Unnamed: 0 int64
ta float64
tr float64
vel float64
rh float64
met float64
clo float64
pmv float64
ppd float64
pmv_ce float64
ppd_ce float64
thermal_acceptability int64
thermal_preference object
thermal_comfort float64
dtype: object
second["thermal_comfort"].value_counts()
5.0 672
6.0 214
4.0 198
3.0 142
2.0 41
1.0 12
0.0 1
Name: thermal_comfort, dtype: int64
[112] second.corr()
Unnamed: 0 ta tr vel rh met clo pmv ppd pmv_ce ppd_ce thermal_acceptability thermal
Unnamed: 0 1.000000 -0.269818 -0.172875 0.139525 0.409950 0.049248 0.214011 -0.047778 0.064101 -0.075340 0.089336 0.097630
ta -0.269818 1.000000 0.962695 0.453734 -0.246542 -0.325632 -0.816464 -0.066888 0.010235 -0.011413 -0.044955 0.022439
tr -0.172875 0.962695 1.000000 0.539354 -0.176448 -0.293345 -0.809658 -0.084426 0.028980 -0.030086 -0.025906 0.038173
vel 0.139525 0.453734 0.539354 1.000000 0.029718 -0.050460 -0.394019 -0.238433 0.215041 -0.229320 0.203423 0.023676
rh 0.409950 -0.246542 -0.176448 0.029718 1.000000 0.211970 0.110780 0.008945 0.005556 -0.000841 0.012536 0.054239
met 0.049248 -0.325632 -0.293345 -0.050460 0.211970 1.000000 0.122179 0.203770 -0.194821 0.147936 -0.144268 -0.015629
Disk 84.44 GB available
0s completed at 11:11 PM


colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
.config
gdrive
sample_data
db_measurements_v2.1.0.csv
one.csv
pmv_model.sav
ppd_model.sav
thermal_acc_model.sav
thermal_comfort_model.sav
three.csv
two.csv
unsup_kMeans_model.sav
[123] second_df = second.drop(0,axis=0)
second_df = second.drop("Unnamed: 0",axis=1)
X = second_df.iloc[:,:10]
y = second_df["thermal_acceptability"]
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# create a logistic regression model
lr = LogisticRegression()
# fit the model with the training data
lr.fit(X_train, y_train)
# make predictions on the testing data
y_pred = lr.predict(X_test)
# evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
Accuracy: 0.8411458333333334
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
[132] from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# create a Random Forest classifier with 100 trees
rf = RandomForestClassifier(n_estimators=100)
Disk 84.44 GB available
0s completed at 11:11 PM


colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
config
gdrive
sample_data
db_measurements_v2.1.0.csv
one.csv
pmv_model.sav
ppd_model.sav
thermal_acc_model.sav
thermal_comfort_model.sav
three.csv
two.csv
unsup_kMeans_model.sav
[132] from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# create a Random Forest classifier with 100 trees
rf = RandomForestClassifier(n_estimators=100)
# fit the model with the training data
rf.fit(X_train, y_train)
# make predictions on the testing data
y_pred = rf.predict(X_test)
# evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
Accuracy: 0.8333333333333334
# save the model to disk
import pickle
filename = 'thermal_acc_model.sav'
pickle.dump(rf, open(filename, 'wb'))
[134] X1 = second_df.iloc[:,:10]
y1 = second_df["thermal_comfort"]
[136] from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3, random_state=42)
# create a logistic regression model
lr1 = LogisticRegression()
# fit the model with the training data
lr1.fit(X_train, y_train)
0s completed at 11:11 PM


colab.research.google.com
PMV_PPD.ipynb
File Edit View Insert Runtime Tools Help All changes saved
[134] X1 = second_df.iloc[:,:10]
y1 = second_df["thermal_comfort"]
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3, random_state=42)
# create a logistic regression model
lr1 = LogisticRegression()
# fit the model with the training data
lr1.fit(X_train, y_train)
# make predictions on the testing data
y_pred = lr1.predict(X_test)
# evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
Accuracy: 0.5
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
# save the model to disk
import pickle
filename = 'thermal_comfort_model.sav'
pickle.dump(lr1, open(filename, 'wb'))
0s completed at 11:11 PM

4 References

https://github.com/sujaykumarmag/pythermalcomfort_ml

https://www.softxjournal.com/action/showPdf?pii=S2352-7110%2820%2930245-4

https://cbe-berkeley.gitbook.io/thermal-comfort-tool/

https://pythermalcomfort.readthedocs.io/en/latest/usage.html