Column	In [1]:	<pre>import numpy as np import pandas as pd import matplotlib.pyplot as plt</pre>
The content of the	In [2]:	<pre>train = pd.read_csv(r"C:\Users\on3_a\Documents\Data Analytics, Big Data & Predictive Analytics/Data/bikeshare_v</pre>
Section Sect		<pre><class 'pandas.core.frame.dataframe'=""> RangeIndex: 8722 entries, 0 to 8721 Data columns (total 21 columns): # Column</class></pre>
The content of the		6 Stn Press (kPa) 8722 non-null float64 7 Hmdx 8722 non-null float64 8 Wind Chill 8722 non-null float64 9 Strong Wind 8722 non-null int64 10 Fog 8722 non-null int64 11 Freezing Rain 8722 non-null int64 12 Haze 8722 non-null int64 13 Heavy Rain 8722 non-null int64 14 Moderate Rain 8722 non-null int64 15 Moderate Snow 8722 non-null int64 16 Rain 8722 non-null int64 17 Snow 8722 non-null int64 18 Thunderstorms 8722 non-null int64 19 Weekend 8722 non-null int64 19 Weekend 8722 non-null int64 20 Holiday 8722 non-null int64 dtypes: float64(7), int64(13), object(1) memory usage: 1.4+ MB
The content of the		Dew Temp Point Rel Visibility Stn Wind Strong Freezing Heavy Moderate Moderate Rain Snow date trips (°C) Temp (%) (km) (kPa) Chill Wind Rain Rain Snow
1		0 01-01 4 -16.2 -20.5 70.0 16.1 102.13 0.0 -22.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
The content of the		01:00:00 2018- 2
Compared to the control of the con		03:00:00 2018- 4 01-01 6 -17.2 -21.1 72.0 16.1 102.09 0.0 -21.0 0 0 0 0 0 0 0 0
Column		# Scale numeric features from sklearn.preprocessing import MinMaxScaler
Compared with a confidence of the confidence o		<pre>train.columns Index(['date', 'trips', 'Temp (°C)', 'Dew Point Temp (°C)', 'Rel Hum (%)',</pre>
Company	In [7]:	'Strong Wind', 'Fog', 'Freezing Rain', 'Haze', 'Heavy Rain', 'Moderate Rain', 'Moderate Snow', 'Rain', 'Snow', 'Thunderstorms', 'Weekend', 'Holiday'], dtype='object')
1 1 1 1 1 1 1 1 1 1		'Visibility (km)', 'Stn Press (kPa)', 'Hmdx', 'Wind Chill'] train[num_vars] = scaler.fit_transform(train[num_vars])
1	Out[8]:	Dew Rel Visibility Stn date trips (°C) Temp (%) (km) (kPa) Stn Wind Strong Freezing Heavy Moderate Moderate Chill Wind Rain Rain Rain Snow
Section 1 - Sectio		0 01-01 00:00:00 4 0.107750 0.139847 0.638554 1.0 0.748614 0.0 0.31250 0 0 0 0 0 0 0 2018- 1 01-01 10 0.096408 0.128352 0.638554 1.0 0.743068 0.0 0.37500 0 0 0 0 0 0
Common C		2018- 2 01-01 6 0.086957 0.134100 0.698795 1.0 0.752311 0.0 0.18750 0 0 0 0 0 0 0 0 0 0 2018-
Second Communication of the		03:00:00 2018- 4 01-01 6 0.088847 0.128352 0.662651 1.0 0.741220 0.0 0.34375 0 0 0 0 0 0
Column C		<pre>X,y =train[['Temp (°C)', 'Dew Point Temp (°C)', 'Rel Hum (%)',</pre>
Linear Regression Model (Trial 1) 20 107 20	In [10]:	'Weekend', 'Holiday']].values,train['trips'].values
Section Proceed of the Community of		<pre>print ('Training Set: %d rows\nTest Set: %d rows' % (X_train.shape[0], X_test.shape[0]))</pre> Training Set: 6105 rows
Comparison of Contention Deposit Cross and process Contention Co		
12 [28]	In [23]:	<pre>from sklearn.model_selection import cross_val_predict</pre>
		<pre>start = time.time() predicted = cross_val_predict(regressor, X_train, y_train, cv=15)</pre>
chi decentrality particularly particularly (Construction) (Constru		<pre>print(f"Runtime of the program is {end - start}") Runtime of the program is 0.0520319938659668 import matplotlib.pyplot as plt</pre>
1987		<pre>ax.scatter(y_train, predicted, edgecolors = (0,0,0)) ax.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'k', lw=4) ax.set_xlabel('Measured') ax.set_ylabel('Predicted')</pre>
		1200 - 1000 -
10 10 10 10 10 10 10 10		600 - 400 - 200 - 0 -
### Security	In [20]:	0 200 400 600 800 1000 1200 1400 Measured
rz: 0.1800/0800001162873 Rooffen 2011577 Rop month orest Regression Model (Trial 1) In [28]: from aktearr.onombio. Amport RandemPorestRagranates In [29]: from aktearr.onombio. Amport RandemPorestRagranates In [29]: from aktearr.onombio. Amport RandemPorestRagranates In [29]: from aktearr.onombio. Amport RandemPorestRagranates In [28]: from aktearr.onombio. Amport RandemPorestRagranates In RandemPorestRagranates In RandemPorestRagranates In RandemPorestRagranates In RandemPorestRagranates In RandemPorestRagranates	In [27]:	<pre>for i in scoring: scores = cross_val_score(regressor, X_train, y_train, cv=15, scoring = i) # print(scores) if i == 'r2': print(i, ': ', scores.mean()) elif i == 'neg_mean_squared_error': x = -1*scores.mean() y = math.sqrt(x) print('RMSE: ', "%0.3f" % y) elif i == 'neg_mean_absolute_error': x = -1*scores.mean()</pre>
In [28]: from skleath.ensemble import DandonTorestRegressor In [28]: rect = %indonTorestRegressorin_natimatorsell, critarione*mac*, rendom_state=2) In [28]: statt = time.time() point(ffNoutine of the program is (end = statl*) Rictime of the program is 10.63499782562259 In [31]: fin as = plut.explore() acception (program is 10.63499782562259) In [31]: fin as = plut.explore() acception (program is 10.63499782562259) In [32]: state = time.time() acception (program is 10.63499782562259) In [32]: for in program (program is 10.63499782562259) In [32]: state = time.time() acception (program is 10.63499782562259) In [32]: state = time.time() acception (program is 10.63499782562259) In [32]: state = time.time() acception (program is 10.63499782562259) In [32]: state = time.time() acception (program is 10.63499782562259) rect = is according to the time is time is the time is the time is the time is time is the time is the time is time is time is time is the time is time		r2: 0.34476860531144393 RMSE: 201.572 neg_mean_absolute_error: 140.18 (+/- 14.38)
<pre>in [38]: start = time.time() predicted = cross_val_predict[rfr], x_trar, y_train, new15) end = time.time() print(("Mentam of the program is (end = start)") Zustime of the program is [0.8499782582599] In [31]: flayar = pit.subplets() start = time.time() start = time.time() In [31]: scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error') scoring = ('12', 'tree mean squared error', 'tree mean absolute error', 'tree mean absolute error' scoring</pre>	In [28]:	<pre>from sklearn.ensemble import RandomForestRegressor</pre>
Runtime of the program is 10.654997825622559 In [31]: fig.ax = pit.subplois() ax.scatter(y_train, predicted, edgecolors = (0.0,01) ax.scatter(y_train, y_train.max()], (y_train.min(), y_train.max()], (k-1, 1e-4) ax.scat_valob(('Yeasured') ax.scat_valob(('Yeasured')) pit.sbox() In [32]: scoring = ('t2', 'neg_mean_squared_error', 'neg_mean_absolute_error') for i in accorning: gooring = ('t2', 'neg_mean_squared_error', 'neg_mean_absolute_error') for i in accorning: gooring = (rone, val_scare(rfit, x_train, y_train, dval5, scoring = i)		<pre>start = time.time() predicted = cross_val_predict(rfr1, X_train, y_train, cv=15) end = time.time()</pre>
In [32]: scoring = (':2', 'neg_mean_squared_error', 'neg_mean_absolute_error') for i in scoring: scores = cross_val score(rfr1, X_train, y_train, cv=15, scoring = i) print(cacres) if i neg_mean_squared_error':		<pre>fig,ax = plt.subplots() ax.scatter(y_train, predicted, edgecolors = (0,0,0)) ax.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'k', lw=4) ax.set_xlabel('Measured') ax.set_ylabel('Predicted')</pre>
In [32]: scoring = ['r2', 'neg_mean_squared_error', 'neg_mean_absolute_error'] for i in scoring: scores = cross_val_score(rfr1, X_train, Y_train, cv=15, scoring = i) #		1200 - 1000 - 1000 - 1000 - 400 - 200 -
<pre>y = math.sqrt(x) print('RMSE: ', "%0.3f" % y) elif i == 'neg_mean_absolute_error': x = -1*scores.mean() print(i, ": %0.2f (+/- %0.2f)" % (x, scores.std() * 2)) r2 : 0.4026057258551218 RMSE: 192.321 neg_mean_absolute_error : 127.79 (+/- 16.21) Gradient Boosting Regression Model (Trial 1) In [33]: from sklearn.ensemble import GradientBoostingRegressor In [34]: gbr = GradientBoostingRegressor(learning_rate = 0.12,</pre>	In [32]:	<pre>scoring = ['r2','neg_mean_squared_error','neg_mean_absolute_error'] for i in scoring: scores = cross_val_score(rfr1, X_train, y_train, cv=15, scoring = i) # print(scores) if i == 'r2': print(i, ': ', scores.mean()) elif i == 'neg_mean_squared_error':</pre>
<pre>RMSE: 192.321 neg_mean_absolute_error : 127.79 (+/- 16.21) Gradient Boosting Regression Model (Trial 1) In [33]:</pre>		<pre>y = math.sqrt(x) print('RMSE: ', "%0.3f" % y) elif i == 'neg_mean_absolute_error': x = -1*scores.mean()</pre>
<pre>In [33]: from sklearn.ensemble import GradientBoostingRegressor In [34]: gbr = GradientBoostingRegressor(learning_rate = 0.12,</pre>		RMSE: 192.321 neg_mean_absolute_error : 127.79 (+/- 16.21)
<pre>max_depth = 8,</pre>	In [33]:	<pre>from sklearn.ensemble import GradientBoostingRegressor gbr = GradientBoostingRegressor(learning_rate = 0.12,</pre>
	In [36]:	<pre>max_depth = 8, min_samples_leaf = 1, random_state = 2)</pre> start = time.time()
<pre>predicted = cross_val_predict(gbr, X_train, y_train, cv=15) end = time.time() print(f"Runtime of the program is {end - start}") Runtime of the program is 19.78090763092041</pre>		<pre>predicted = cross_val_predict(gbr, X_train, y_train, cv=15) end = time.time() print(f"Runtime of the program is {end - start}")</pre>
<pre>fig,ax = plt.subplots() ax.scatter(y_train, predicted, edgecolors = (0,0,0)) ax.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'k', lw=4) ax.set_xlabel('Measured') ax.set_ylabel('Predicted') plt.show()</pre>	In [37]:	<pre>ax.scatter(y_train, predicted, edgecolors = (0,0,0)) ax.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'k', lw=4) ax.set_xlabel('Measured') ax.set_ylabel('Predicted') plt.show()</pre>
		1000 -
<pre>Measured In [38]: scoring = ['r2', 'neg_mean_squared_error', 'neg_mean_absolute_error'] for i in scoring: scores = cross_val_score(gbr, X_train, y_train, cv=15, scoring = i) # print(scores) if i == 'r2': print(i, ': ', scores.mean()) elif i == 'neg_mean_squared_error': x = -1*scores.mean() y = math.sqrt(x) print('RMSE: ', "%0.3f" % y) elif i == 'neg_mean_absolute_error':</pre>	In [38]:	<pre>Measured scoring = ['r2','neg_mean_squared_error','neg_mean_absolute_error'] for i in scoring: scores = cross_val_score(gbr, X_train, y_train, cv=15, scoring = i) # print(scores) if i == 'r2': print(i, ': ', scores.mean()) elif i == 'neg_mean_squared_error': x = -1*scores.mean() y = math.sqrt(x) print('RMSE: ', "%0.3f" % y) elif i == 'neg_mean_absolute_error':</pre>
<pre>x = -1*scores.mean() print(i, ": %0.2f (+/- %0.2f)" % (x, scores.std() * 2)) r2 : 0.36474598912547357 RMSE: 198.333 neg_mean_absolute_error : 132.01 (+/- 16.23)</pre>		<pre>print(i, ": %0.2f (+/- %0.2f)" % (x, scores.std() * 2)) r2 : 0.36474598912547357 RMSE: 198.333</pre>