Project Report For Autumn Semester 3^{rd} year

Satvik Verma* Mukul Gupta†

April 2023

1 Acknowledgement

We would like to thank Prof Uttam Kumar and Rahisha Thottolil for providing us this opportunity to work on this project through which we learned a lot and had a chance to work on a real life implementation of what we are studying in the college.

2 Introduction

In this project we took the area and road network data and tried to establish a correlation between them and to prove that where there is more urban buildings, the road network is also more intense and bigger. In this project we also tried to predict the road network for new cites by using the data for 505 places of equal area with unequal topology and land use and land cover to train the machine learning model so it can accurately predict the road network for any topology and land use land cover. This lead to the same conclusion as the correlational matrix that **more urban development leads to more intense road network**.

3 Data preparation

3.1 Getting the tif images

1. We were given a vector file containing 505 vector in it, on which we used vector split function of **QGis** software which in turn gave use 505 different vector files sorted according to the grid number.
2. Then we used google earth engine where we manually uploaded each of the 505 vector files we got and used the **European Space Agency (ESA) WorldCover 10 m 2020** map for the rastor extraction.

*Satvik.Verma@iiitb.ac.in
†Mukul.Gupta@iiitb.ac.in

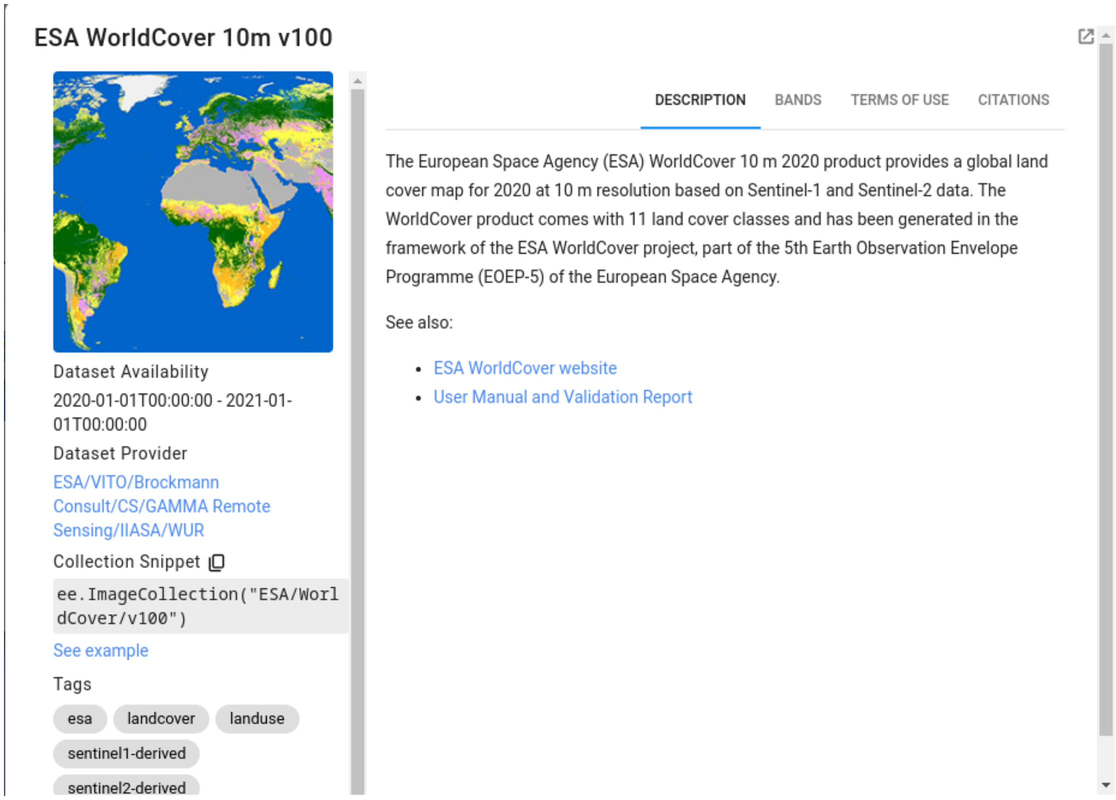

Figure 1: Earth Engine page of European Space Agency (ESA) WorldCover 10 m 2020

3. Then we used the clip function for clipping the rastor out of the above said map with the vector that is to be used.

4. Then we used the *Export.image.toDrive* function of google earth engine to export each image to Google Drive.

5. This process was repeated 505 times.

Some of the resulting images are:-

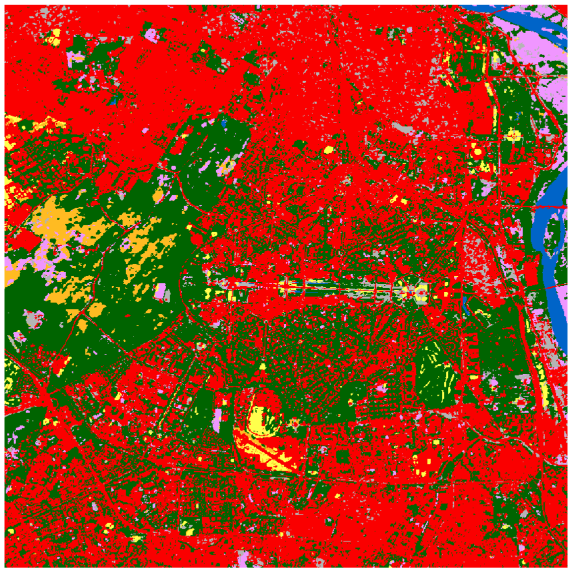

Figure 2: GridNumber_1

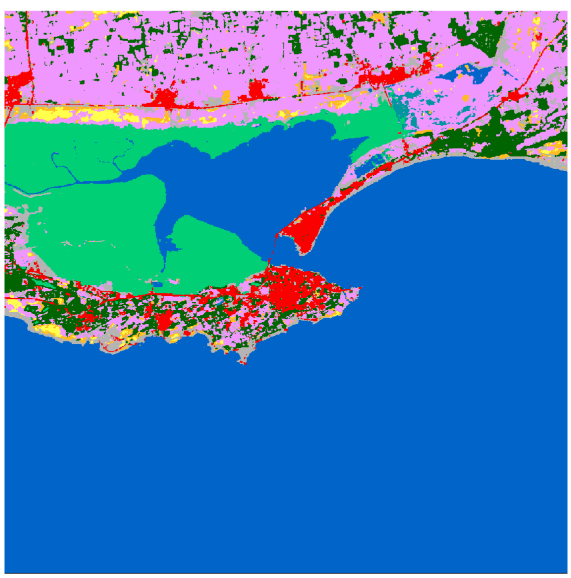

Figure 3: GridNumber_331

Here the coloring encoding the the default as from the European Space Agency (ESA), that is gvien below:-

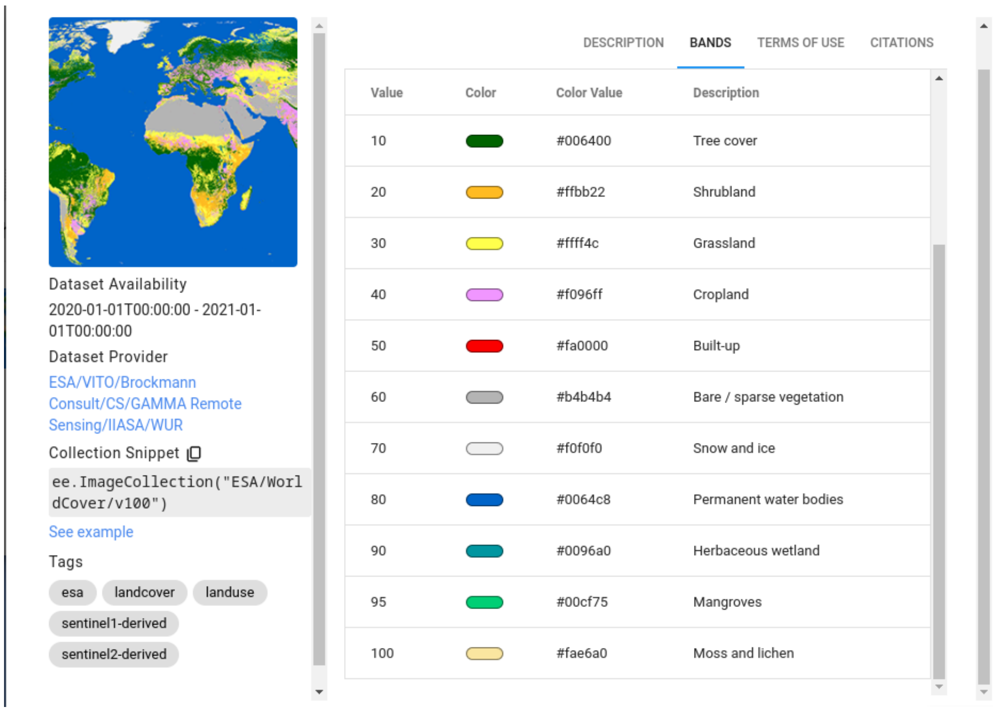

Figure 4: Coloring encoding for the rastor images

3.2 Getting pixel sum and percentages from rastor images

In order to apply the machine learning model we needed the pixel values and the percentages for each band for each of the 505 rastor images. For this purpose we used **Semi-Automatic Classification Plugin** in qgis. This plugin helps in the getting the band information for rastor images.

1. First we uploaded the image that we needed the values for.
2. Then going to Post Processing-¿Classification report, the csv file for the given image was generated and stored in a before mentioned location in the local system.
3. This process was repeated 505 times, as each image has to be processed seperately.

Class	PixelSum	Percentage %
10	453517	36.5118226244052
20	23845	1.91971725531555
30	17686	1.42386745135294
40	38913	3.13281432401317
50	648856	52.2382075661576
60	43025	3.46386390899357
80	16267	1.30962636159438
90	1	8.05081675535983E-05

Figure 5: Pixel Sum and percentage for GridNumber_1

Class	PixelSum	Percentage %
0	1115	0.0896860986547085
10	63304	5.09191819662571
20	13533	1.08853988618311
30	8154	0.655874841641698
40	195152	15.6972390355728
50	36852	2.96422610549177
60	42239	3.39753463773653
80	753140	60.5795411128316
90	3698	0.29745219087454
95	126038	10.1379878943876

Figure 6: Pixel Sum and percentage for GridNumber_331

3.3 Converting all 505 csv files to 1 file

Now after getting all the values in 505 file, we had to add them to a single csv file that has all grid values as rows with the band as columns.
This wasn't possible on the initial csv files as some files had abundance of some classes and others didn't have them, when they didn't have that particular band SCP jsut skipped the band instead of writing 0, for example GridNumber_1 didn't have band 95 (Mangroves) at all but GridNumber_331 had more than 10

percent of its area for band 95. So in order to make for these edge cases we had to write a python script[1] to add all classes to all the 505 files and to add pixelSum and Percentage as 0 for them, as show below:-

Class	PixelSum	Percentage %
0	0	0
10	453517	36.5118226244052
20	23845	1.91971725531555
30	17686	1.42386745135294
40	38913	3.13281432401317
50	648856	52.2382075661576
60	43025	3.46386390899357
70	0	0
80	16267	1.30962636159438
90	1	8.05081675535983E-05
95	0	0
100	0	0

Figure 7: Modified Pixel Sum and percentage for GridNumber_1

Class	PixelSum	Percentage %
0	1115	0.0896860986547085
10	63304	5.09191819662571
20	13533	1.08853988618311
30	8154	0.655874841641698
40	195152	15.6972390355728
50	36852	2.96422610549177
60	42239	3.39753463773653
70	0	0
80	753140	60.5795411128316
90	3698	0.29745219087454
95	126038	10.1379878943876
100	0	0

Figure 8: Modified Pixel Sum and percentage for GridNumber_331

[1] To access the code please contact the author

Now we have to add all value to their corresponding rows and columns in one csv file. This was done by another python script using pandas and numpy module. The following is a apart of the so formed csv file:-

Number	Pixel 0		Pixel 10 (Tree Cover)		Pixel 20 (Shrubland)		Pixel 30 (Grassland)		Pixel 40 (Cropland)		Pixel 50 (Built-up)	
	PixelSur	Percentage %	PixelSum	Percentage %	PixelSum	Percentage %	PixelSum	Percentage %	PixelSum	Percentage %	PixelSum	Percentage %
1	0	0	453517	36.5118226244052	23845	1.91971725531555	17686	1.42386745135294	38913	3.13281432401317	648856	52.2382075661576
2	1114	0.0896860986547085	192492	15.4971781887272	94349	7.59586510051445	82489	6.64103823332877	503738	40.5550233071145	280820	22.6083036124015
3	0	0	21442	1.72780573023604	29779	2.39960483353693	27115	2.18493854935874	571106	46.019971055507	416053	33.5257325567528
4	0	0	47573	3.83001505502733	42908	3.4544444533898	5440	0.437964431491575	526983	42.4264356618979	345305	27.7998727970953
5	0	0	33283	2.68195868479834	96791	7.79946107803732	78812	6.35070540114553	577072	46.5007139426719	287796	23.1907274479531
6	0	0	76516	6.16569271778475	2522	0.203223862123649	81571	6.57302682683909	773595	62.3366231639748	196982	15.8728956418876
7	0	0	33445	2.6925956638301	28128	2.26453373694761	41650	3.35316517860737	774086	62.3202453888947	287395	23.1376448140664
8	0	0	43111	3.47078761140318	86157	6.93634219191537	18878	1.51983318707683	607396	48.9003389393854	258455	20.8077384450653
9	1	8.05081675535983E-05	291088	23.4349614768418	96944	7.80478379531604	129728	10.4441635603932	165572	13.3298983181844	424363	34.1646875075476
10	0	0	36857	2.96995316665001	53667	4.32451031268433	25250	2.03465603434661	729911	58.8165473538996	271286	21.8603444330199
11	0	0	36857	2.96995316665001	53667	4.32451031268433	25250	2.03465603434661	729911	58.8165473538996	271286	21.8603444330199
12	1114	0.0896860986547085	82889	6.67324150035021	78954	6.3564418610268	39026	3.14191174694673	167362	13.4740079381053	303178	24.4083052225648
13	0	0	394240	31.7680314843884	76	0.0061241132122907	6963	0.56108158285764	2260	0.182111787628647	119983	9.66828257303005
14	0	0	39977	3.22136413010195	12355	0.99557129918227	32531	2.62136219617146	871886	70.2569548975178	240509	19.3803203233532
15	0	0	134017	10.7991484259417	185	0.0149073808457078	14699	1.18445184351924	696654	56.1366837604634	172831	13.9267975078082
16	0	0	448113	36.1091413670955	54	0.0043513435982066	7554	0.608704621126901	4513	0.363659512198267	163108	13.14331391882
17	1114	0.0896860986547085	20623	1.66031993945786	57728	4.64757549653412	2924	0.235405881926722	701019	56.4377551102559	332018	26.7301607748106
18	0	0	480050	38.6826387836867	9375	0.755441596910868	10938	0.881388819947848	242379	19.5310057405503	286739	23.1055539260401
19	0	0	150788	12.1505629349329	7027	0.566238730825885	31511	2.53917015042756	596272	48.0478583331453	214063	17.24929008635
20	0	0	62996	5.07624520949302	37402	3.0138695048171	238973	19.2565487721153	379207	30.5566657749098	352027	28.3664894971458
21	0	0	58287	4.696791931642	30128	2.42772740605127	132786	10.69995390799	716606	57.7444246395637	230366	18.5629929508234
22	0	0	346468	27.9185428478416	2202	0.177438122282425	10281	0.828447472836335	83314	6.71347852853676	554065	44.6467998285248
23	1114	0.0896860986547085	25148	2.02461939763789	30309	2.44012205038201	7027	0.565730893399135	931682	75.0080105626716	180490	14.530919161749
24	0	0	28832	2.32121148690535	11594	0.933411694616419	2098	0.168906135527449	727566	58.5750054343013	324996	26.1648324222492
25	1114	0.0896860986547085	12560	1.01118258447319	20394	1.64188356908808	17113	1.37773627134473	884393	71.2008598272295	263805	21.238457141477
26	0	0	73731	5.93594770189436	68340	5.50192817061291	133758	10.7686114756342	720012	57.9668467366014	163022	13.1246024909227
27	1114	0.0896860986547085	45116	3.63220648734814	114981	9.25690961348029	67941	5.46980541175902	610512	49.1512023894824	254165	20.4623584062603
28	0	0	322733	25.9826424390754	5331	0.429189041228233	39926	3.21436909774497	199621	16.0711209152169	221543	17.8360209643268
29	0	0	97506	7.85707609049505	132331	10.6632898091533	36513	2.9422334963207	695119	56.0129927896625	188350	15.1773253096706
30	0	0	210506	16.9626654719274	192	0.0154714439047345	7112	0.57308806797121	505443	40.7288178205248	402850	32.4618290469913
31	0	0	60869	4.90485061998588	302960	24.4126491946791	55715	4.48953904766816	356108	28.6953382605585	370765	29.8764057257235
32	1114	0.0896860986547085	151329	12.1832204877185	1286	0.103533503473927	110427	8.8902754184412	751158	60.4743541232258	176260	14.1903696129972
33	1114	0.0896860986547085	317890	25.5927413836134	1798	0.14475368526137	10516	0.84662388999364	74192	5.97306196713657	595071	47.9080757742873
34	0	0	82967	6.68551711689643	644	0.0518938014304639	12695	1.02296864776357	567295	45.7128790100855	421486	33.9635260709946
35	1114	0.0896860986547085	229768	18.4982006424552	22419	1.80491260838412	20089	1.61732857798424	469746	37.8183896756326	419043	33.7363840561625
36	0	0	116401	9.3712312114064	45272	3.6447657614865	208864	16.8152579079148	576299	46.3967764529712	133172	10.7214336894478
37	0	0	27540	2.21918523508537	28279	2.27873417803119	18431	1.48517803441752	568084	45.7764569748815	320673	25.8399704753279
38	0	0	95849	7.71662735184485	280	0.0225422869150075	13472	1.08460603328208	756217	60.8816449428795	225926	18.1888882627143
39	0	0	401976	32.3914017450499	276291	22.2636495202241	8610	0.693797562602942	308738	24.8782429596872	209723	16.8995709897534
40	0	0	703757	56.7090466045015	520	0.0419018272419895	57719	4.65102224342383	218034	17.5692750016922	218859	17.6357538622203
41	0	0	199360	16.0501082834854	466	0.0375168060799768	20305	1.63471834217581	682270	54.9283074767935	196087	15.7866050510824
42	0	0	146689	11.8202637236542	1169	0.0941985308574725	8618	0.694442206098972	823723	66.3759593101025	207131	16.6907064970395
43	0	0	264414	21.3065956699296	12884	1.03819835035729	165897	13.3680527576237	455635	36.7152674142382	148545	11.9698210147333

Figure 9: Complete csv file

3.4 Getting Area

We got area for each class in each grip using the formula:

$$area = \frac{PixelSum \times 100}{10000}$$

Since each grid has has dimensions 10m×10m that means each pixel has area of $100m^2$, and dividing by 10000 gives the area in hectare. This area is used as the X values in training and testing the models.

3.5 Target values

We are trying to predict various urban development parameters using the urban land use and land cover data. For this purpose we are using four urban parameters:

- Road Length
- Network Density
- CA
- Population

3.6 Problems Faced

The following are the problems that we faced during the preparation of the data:-

1. The initial map provided of India was in 13 segments with each segment extremely detailed, so merging them in creating one LULC map of India was not possible in our local system.

2. So we tried to run them on Colab and Kaggle after writing Python code for merging but then too the process was exceeding the RAM limit of these tools. So to bypass these problems we switched to Google Earth Engine.

4 Data Analysis

We formed Pearson correlational as given below:

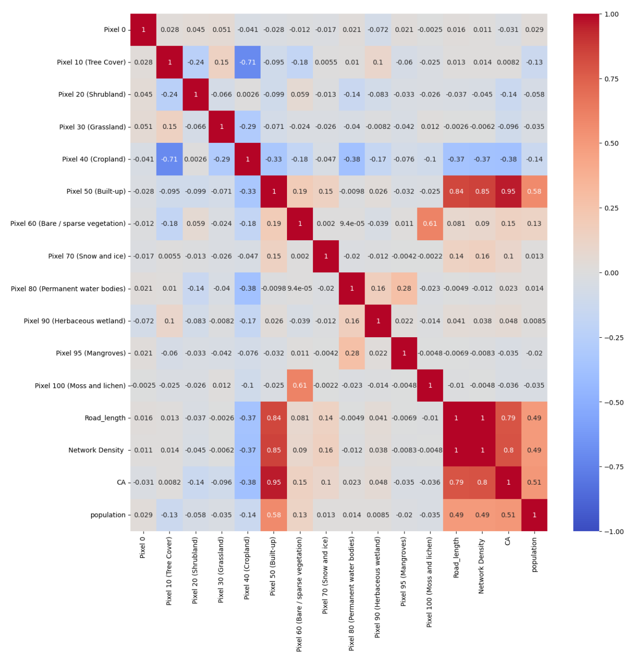

Figure 10: Pearson correlation matrix

It clearly shows that Network density is most highly correlated to Class 50 (Urban Area) among all classes which makes sense as Road Density depends on the urban development of an area.

So based on this matrix we initially used class 50 in all the models.

5 Machine Learning

For the purpose of this project we only used one class from Urban LULC as X and Network Density for Y. From the covariance matrix as shown below the covariance between Pixel 50 (Built-up) adn Network Density is the highest so that was chosen as the preferred value for X.

	Road_length	Network Density	CA	population
Pixel 0	0.0162749151812646	0.0113412254890206	-0.0306972723748172	0.028991100479785
Pixel 10 (Tree Cover)	0.013275942035263	0.013797801674386	0.00817946852924444	-0.131842402949125
Pixel 20 (Shrubland)	-0.036863853073752	-0.0447103652166605	-0.140446387687758	-0.0584455276682268
Pixel 30 (Grassland)	-0.00261475941328234	-0.00622948850879529	-0.0961947575867365	-0.0346062656037743
Pixel 40 (Cropland)	-0.365610090602581	-0.367989030487007	-0.382604630142377	-0.140485459299882
Pixel 50 (Built-up)	0.836750733262049	0.851642245840642	0.954586350200086	0.575680651416894
Pixel 60 (Bare / sparse vegetation)	0.0812099116296624	0.0897391334910739	0.146998116718757	0.125604239927409
Pixel 70 (Snow and ice)	0.140271511402739	0.155859779889916	0.101260009243038	0.0125682043998666
Pixel 80 (Permanent water bodies)	-0.00487365121894222	-0.0118106822581467	0.0230698823386452	0.0136875495812016
Pixel 90 (Herbaceous wetland)	0.0410023015146928	0.037852894639696	0.0478593104187464	0.00851109151261058
Pixel 95 (Mangroves)	-0.00690158322090543	-0.00825866724650441	-0.0351317328154062	-0.0196036599127789
Pixel 100 (Moss and lichen)	-0.0100573751853033	-0.00480557307307805	-0.0357919464918326	-0.0354638985554314

Figure 11: Covariance matrix

5.1 Models Used

We have used the following models on the data that we prepared.

- Linear Regression
- Ridge Regression
- Lasso Regression
- Kernel Regression
- Decision Tree
- Random Forest
- SVM
- Multilayer Preceptron
- Gradient Boost
- XG Boost

5.2 Accuracies and Scores

5.2.1 Model 1

In model 1 we have used Urban Area as X value and predicted Road Network Density, using all the models above listed. The following are the accuracy scores that we got.

	MAE	MSE	R2 Score	Adj R2 SCORE
Linear Regression	1.50788995394405	4.1045736340028	0.772084885700458	0.596115070727089
Ridge Regression	1.50788995482987	4.1045736404745	0.772084885341103	0.596115070172184
Lasso Regression	1.50805318993905	4.1057672425049	0.772018608069102	0.596012731204954
Kernel Ridge	1.51902887118413	4.2301559481055	0.765111662651026	0.585395856324617
Gradient Boost	1.63904724037798	4.96081429342282	0.724540315871576	0.524958669323284
XG Boost	1.62493499574826	5.27390584400885	0.707155246702176	0.500068542938415
Decision Tree(max_depth=10)	1.71800575169953	5.48990685668159	0.695161334573448	0.483249281085937
Random Forest(n_estimators=1000)	1.77744433262184	6.02638575963519	0.66537221117387	0.429893925588867
SVM(kernel=rbf, degree=4)	1.69184566960589	7.35595013790728	0.591545342840418	0.610365359610062
MLP(layers=(2, 1), activation=tanh, solver=sgd)	2.87341212604716	18.6286178712199	-0.03439332557937	0.0013192811267775

Figure 12: Errors for various models with only Urban Area as X

From looking at these accuracies, we can say that **Linear Regression** is best fitting the data as it has the best ratio of R^2 score and Mean Absolute Error.

5.2.2 Model 2

In model 2 we have used all the 11 classes to predict the Road Network Density. The following are the accuracy scores for that.

	MAE	MSE	R2 Score	Adj R2 SCORE
Linear Regression	1.42855899592528	3.72006210420763	0.79343569995721	0.629540209966588
Ridge Regression	1.43071371137232	3.71009554498926	0.793989114193615	0.630418713457961
Lasso Regression	1.4523620821523	3.7758562064437	0.790337614669367	0.624633545161264
Kernel Ridge	1.42987433145426	3.71224206513033	0.793869924132254	0.63022945644175
Gradient Boost	2.85892472162454	18.7490198208559	-0.04107889795792	0.0016874758574368
XG Boost	1.53372834984179	4.5936356562198	0.718635732072319	0.516437315411117
Decision Tree(max_depth=10)	1.93623303675468	7.62444991690088	0.576636325905801	0.332509452354142
Random Forest(n_estimators=100)	1.48417279003446	4.08467698272028	0.773189689257559	0.597822295574201
SVM(kernel=rbf, degree=50)	5.6272446142609	51.3952743940852	-1.85383108758143	3.43668970128335
MLP(layers=(8, 6, 2, 1), activation=tanh, solver=sgd)	2.85892472162454	18.7490198208559	-0.04107889795792	0.0016874758574368

Figure 13: Errors for various models with all classes as X

From looking at these accuracies, we can say that **Linear Regression** is best fitting the data as it has the best ratio of R^2 score and Mean Absolute Error.

5.2.3 Model 3

In model 3 we removed Class 70 (Snow and ice) and Class 100 (Moss and Lichens) from X and ran all the models and got the following accuracy scores.

	MAE	MSE	R2 Score	Adj R2 SCORE
Linear Regression	1.45941675336243	3.82702251409438	0.787496497443479	0.620150733485747
Ridge Regression	1.45941335558735	3.8270099717369	0.78749719388435	0.620151830375726
Lasso Regression	1.4523620821523	3.7758562064437	0.790337614669367	0.624633545161265
Kernel Ridge	1.45786041037772	3.82953339865102	0.787357075278901	0.619931163991745
Gradient Boost	1.41782968482319	3.94843904720129	0.780754588176297	0.609577726958339
XG Boost	1.57700956228922	5.06715124723424	0.718635732072319	0.516437315411117
Decision Tree(max_depth=15)	1.83969486439373	6.53265960867042	0.637260287145062	0.406100673572207
Random Forest(n_estimators=100)	1.48844268947475	4.08944180403232	0.772925112497393	0.597413229529107
SVM(kernel=rbf, degree=50)	3.9590297287635	24.3486630618411	-0.35201090773678	0.123911679165668
MLP(layers=(8, 6, 2, 1), activation=tanh, solver=sgd)	2.90642634608128	18.4051427943236	-0.02198440025959	0.0004833138547737

Figure 14: Errors for various models with all classes except 70 and 100 as X

From looking at these accuracies, we can say that **Linear Regression** is best fitting the data as it has the best ratio of R^2 score and Mean Absolute Error.

6 Conclusion

From the low error rate (Mean Absolute Error) in Linear Regression, Ridge Regression, Lasso Regression in test data we can say that Network Density and Urban Growth have almost Linear Relation. Also even after using all the parameters and after removing targetted parameters we can see that the difference in accuracy scores is not that great from just using Urban Growth in model 1, so we can say that urban growth is playing the major role which is also proved by the correlation matrix.