Anirudh Rao
BE21B004

DA5401 – Data Analytics Lab

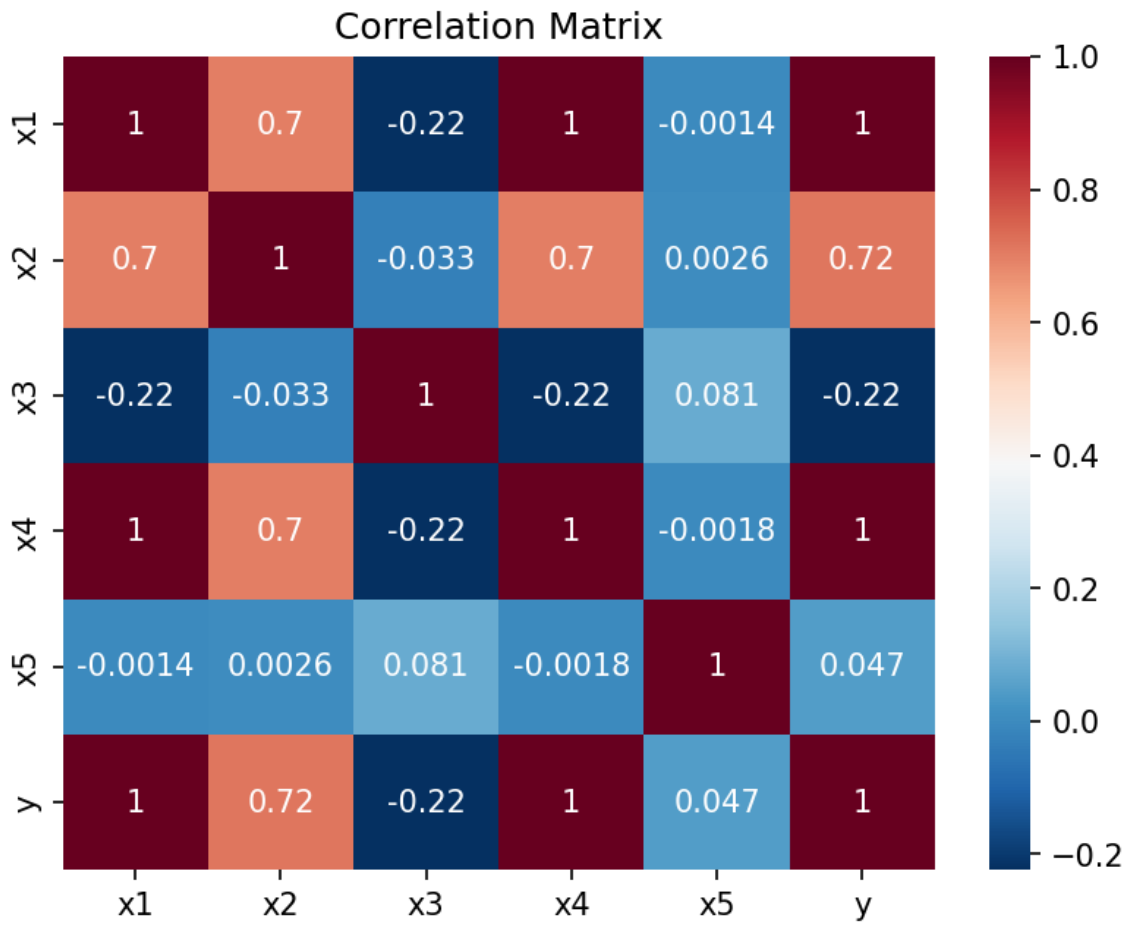
Assignment 3

Task 1

sklearn's `LinearRegression` class was used to fit OLS on the given data in order to predict the label y from the features x_1, x_2, x_3, x_4, x_5. The loss, in terms of SSE was found to be **71877.84**.

Task 2

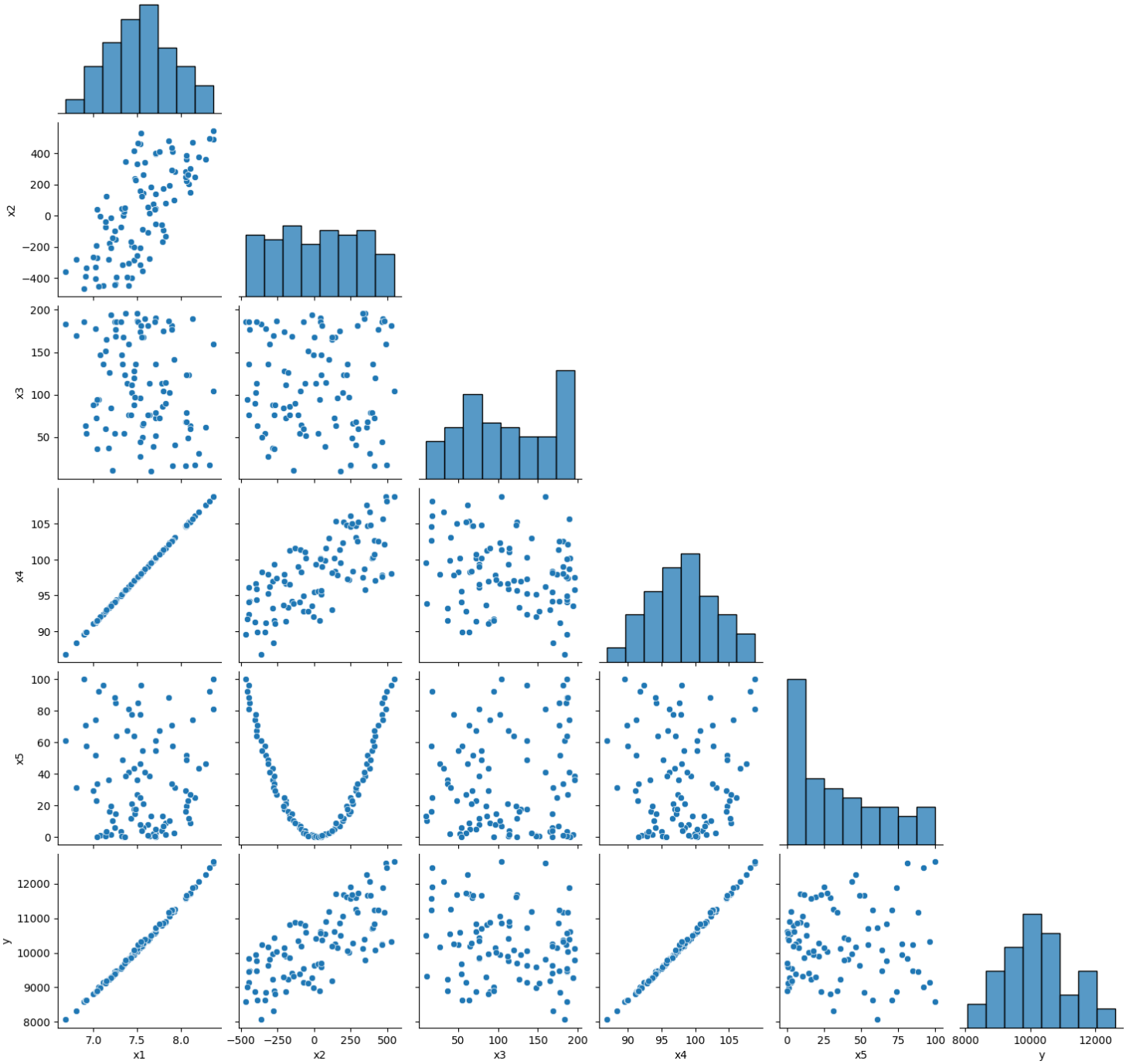
To study how the features interact with each other and with the label, their pairwise correlations were computed and visualised as a heatmap.



From this, we can observe the following:

- x_1 and x_4 are highly correlated with each other.
- x_2 has a moderately strong correlation with x_1 and x_4.
- The label y has a strong correlation with x_1 and x_4, and a moderately strong correlation with x_2.
- x_3 and x_5 seem to be uncorrelated with any of the features or with the label.

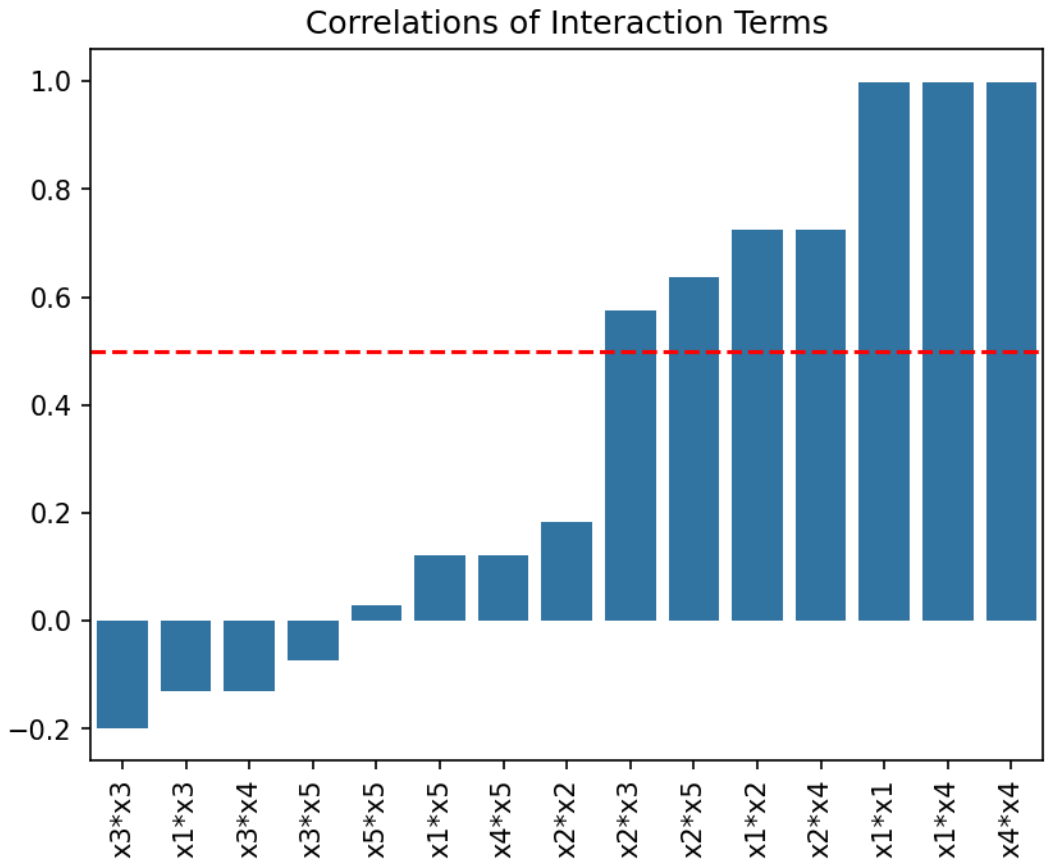
To further analyse the relationships between the features and the label, we use a pairplot.



From this, we can observe the following:

- x_1 has a fully linear relationship with x_4, and a noisy linear relationship with x_2.
- x_2 has a quadratic relationship with x_5.
- x_3 does not seem to have a direct relationship with any of the features or with the label.
- y has a fully linear relationship with x_1 and x_4, and a noisy linear relationship with x_2.

Lastly, we check whether the interactions between the features in a higher dimensional space have any effect on the label y.



The interaction terms x_2x_3, x_2x_5, x_1x_2, x_2x_4, x_1^2, x_1x_4, and x_4^2 have correlations greater than 0.5 with the label y.

Based on our exploratory data analysis, we can consider the following feature engineering steps:

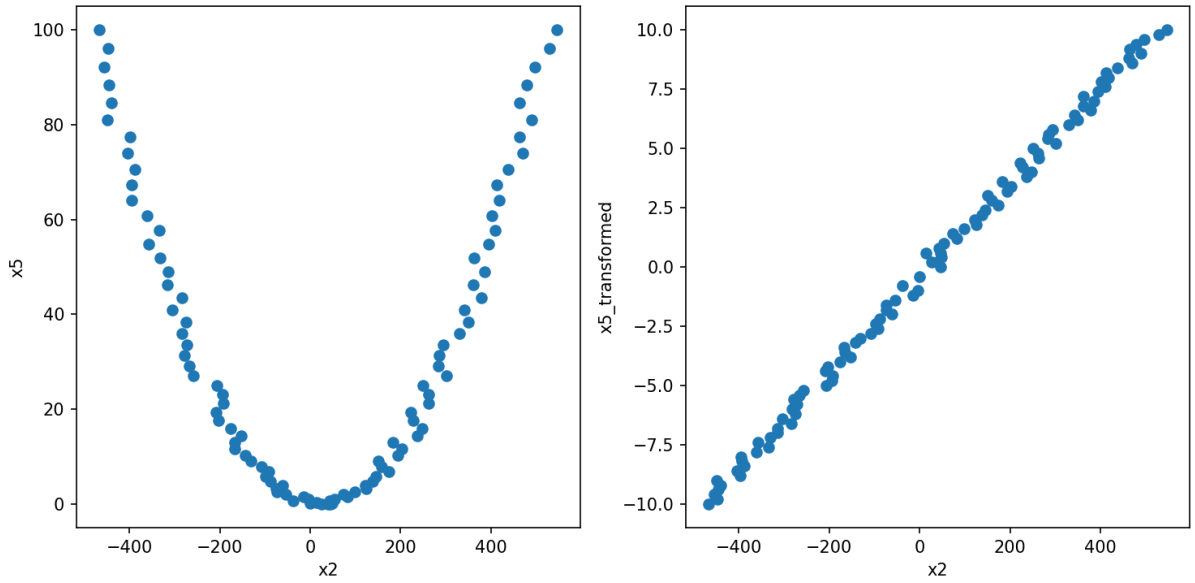
- Exclude x_1 as a feature as it is highly correlated with x_4.
- Exclude x_3 as a feature as it is non-informative on its own.
- Include the interaction terms x_2x_3, x_2x_5, x_1x_2, x_2x_4, x_1^2, x_1x_4, and x_4^2.
- Due to its quadratic relationship with x_2, the feature x_5 can be transformed as follows:

$$(x_5)_{\text{transformed},i} = \text{sgn}(x_{2,i}) \times \sqrt{x_{5,i}}$$

$\text{sgn}(x_{2,i})$ ensures that the transformed x_5 has a linear relationship with x_2.

The transformed x_5 now has a correlation of 0.72 with y, which is a significant increase from its earlier value of 0.047.

The overall transformation looks like:



Task 3

sklearn's `LinearRegression` class was used to fit OLS on the engineered features. The loss, in terms of SSE was found to have been drastically reduced to **36.27**. This is nearly a 2000-fold reduction in SSE.

Task 4

The `LazyRegressor` class from the `lazypredict` library was used to predict the label *y* from the original data and from the engineered data.

The performance on the original data was found to be:

Model	R-Squared	RMSE	SSE
DecisionTreeRegressor	1	0	0
ExtraTreeRegressor	1	0	0
ExtraTreesRegressor	1	1.51E-11	2.32E-20
GaussianProcessRegressor	1	2.65E-06	7.11E-10
XGBRegressor	1	0.010077138	0.010256
GradientBoostingRegressor	0.999982358	4.274553413	1845.452
Lars	0.999312865	26.67699	71877.84
RANSACRegressor	0.999312865	26.67699	71877.84
LinearRegression	0.999312865	26.67699	71877.84
TransformedTargetRegressor	0.999312865	26.67699	71877.84
OrthogonalMatchingPursuitCV	0.999306611	26.79811206	72532.02
LassoLarsIC	0.999306611	26.79811581	72532.04
LassoLarsCV	0.99930661	26.79812647	72532.1
LassoLars	0.999303599	26.85624885	72847.07
BayesianRidge	0.999296571	26.9914141	73582.18
RidgeCV	0.999282493	27.26017797	75054.85
SGDRegressor	0.999281291	27.28300163	75180.58
HuberRegressor	0.999261753	27.65135194	77224.32
Ridge	0.999237834	28.09574185	79726.44
Lasso	0.999235213	28.14400996	80000.61
LassoCV	0.999235109	28.1459183	80011.46
PoissonRegressor	0.999204178	28.70937049	83247.02
RandomForestRegressor	0.999154228	29.59663656	88472.05
PassiveAggressiveRegressor	0.999081838	30.83722627	96044.39
BaggingRegressor	0.998925967	33.35219864	112349.3
OrthogonalMatchingPursuit	0.996535058	59.9050812	362450.5
AdaBoostRegressor	0.995542911	67.94246326	466234
LarsCV	0.991375941	94.50863905	902120.2
KNeighborsRegressor	0.969293039	178.3339652	3212103
HistGradientBoostingRegressor	0.965868075	188.0165311	3570372
LGBMRegressor	0.964161834	192.6586593	3748853
ElasticNet	0.960639092	201.9055244	4117350
GammaRegressor	0.904539799	314.4320585	9985619
TweedieRegressor	0.904504113	314.4908255	9989352
ElasticNetCV	0.902220645	318.2286179	10228215
SVR	0.018055299	1008.461128	1.03E+08
NuSVR	0.009678435	1012.75353	1.04E+08
DummyRegressor	0	1017.690329	1.05E+08
LinearSVR	-99.34384673	10194.38475	1.05E+10
KernelRidge	-100.3328224	10244.49882	1.06E+10
MLPRegressor	-101.1236553	10284.39676	1.07E+10

The performance on the engineered data was found to be:

Model	R-Squared	RMSE	SSE
DecisionTreeRegressor	1	0	0
ExtraTreeRegressor	1	0	0
ExtraTreesRegressor	1	1.51E-11	2.32E-20
GaussianProcessRegressor	1	2.09E-06	4.4E-10
XGBRegressor	1	0.021459581	0.046512
RANSACRegressor	0.999999653	0.599286597	36.27359
TransformedTargetRegressor	0.999999653	0.599286597	36.27359
LinearRegression	0.999999653	0.599286597	36.27359
BayesianRidge	0.999999652	0.600625962	36.43591
LassoLarsIC	0.999999645	0.606410309	37.14108
LassoLarsCV	0.999999635	0.614577612	38.14827
OrthogonalMatchingPursuitCV	0.999999085	0.973302583	95.67911
HuberRegressor	0.999998672	1.172636807	138.8828
LassoLars	0.999997246	1.688921623	288.0981
RidgeCV	0.999984078	4.060884983	1665.569
GradientBoostingRegressor	0.999979165	4.645289708	2179.45
SGDRegressor	0.999965869	5.945539911	3570.294
Ridge	0.999957953	6.599064117	4398.312
PoissonRegressor	0.999933927	8.272343373	6911.598
PassiveAggressiveRegressor	0.999919592	9.125679687	8411.081
Lasso	0.999848035	12.54548819	15896.32
LassoCV	0.999847985	12.5475322	15901.5
Lars	0.999407335	24.7753742	61995.74
RandomForestRegressor	0.999109297	30.37261331	93172.06
BaggingRegressor	0.99892414	33.38054825	112540.4
OrthogonalMatchingPursuit	0.998108714	44.25824669	197838
AdaBoostRegressor	0.996893711	56.72004926	324933.6
ElasticNet	0.983574122	130.4307367	1718230
KNeighborsRegressor	0.978430486	149.4637333	2256280
HistGradientBoostingRegressor	0.962526019	197.0065042	3919968
LGBMRegressor	0.961280771	200.2529845	4050227
GammaRegressor	0.95604873	213.3542811	4597525
TweedieRegressor	0.955943855	213.6086789	4608495
ElasticNetCV	0.954758277	216.4637679	4732513
LarsCV	0.343658308	824.4809845	68656658
SVR	0.023251427	1005.789367	1.02E+08
NuSVR	0.013847212	1010.61968	1.03E+08
DummyRegressor	0	1017.690329	1.05E+08
LinearSVR	-99.34384673	10194.38475	1.05E+10
KernelRidge	-100.3321023	10244.46242	1.06E+10
MLPRegressor	-100.8141822	10268.80214	1.07E+10

- Most models seem to have poorer performance than OLS.
- However, there are 5-6 models that achieve better SSE than OLS on both the original and transformed data, with some even achieving SSE of 0.
- The variation in performance arises from the models' underlying assumptions of the data.
- For example, linear models assume a linear relationship between the features and the label but other models may not make such an assumption. Thus, the different models adopt different approaches while learning from the data.
- Another source of variation is the models' inherent complexity. More complex models tend to overfit to the training data, minimising the SSE.

Along with this report, the source code has been attached as a Python notebook (.ipynb). This was not attached as a Python script (.py) as the `lazypredict` library was giving errors when run from a script.