Krish Kabra, 404593821
Ranjana Vittal, 605629219
Zhiyuan Chen, 605363281

Project 4: Regression Analysis

EE 219 - Large-Scale Data Mining: Models and Algorithms
Winter 2021

2021-03-22

Data Inspection

Question 1: Plot a heatmap of Pearson correlation matrix of dataset columns.

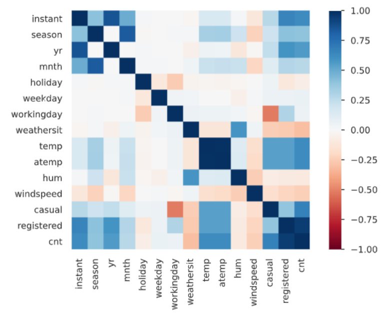


Figure 1: Pearson correlation matrix for the bike sharing dataset.

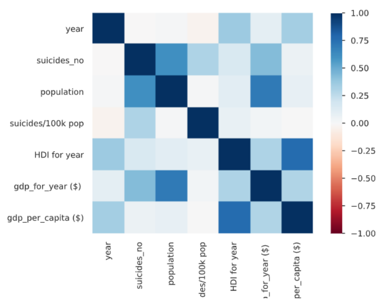


Figure 2: Pearson correlation matrix for the suicide rates overview dataset.

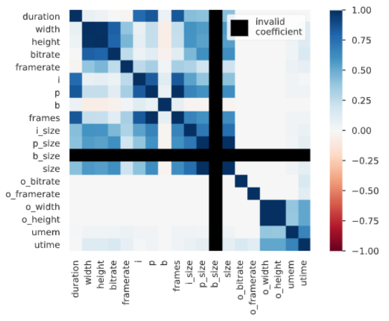


Figure 3: Pearson correlation matrix for the video transcoding time dataset. The `b_size` feature is invalid as all instances were 0 in the dataset.

Bike Sharing Dataset: Figure 1 shows the heatmap of Pearson correlation matrix for the bike sharing dataset columns. The `temp` and `atemp` features have the highest absolute correlation with the target variables. This implies that warmer real and feel temperature for a given day largely influences people to rent bikes.

Suicide Rates Overview Dataset: Figure 2 shows the heatmap of Pearson correlation matrix for the suicide rates overview dataset columns. The `population` and `gdp_for_year` features have the highest absolute correlation with the total suicide number. Intuitively, it makes sense for a larger population to have a larger total number of suicides. However, we see that the suicide rate (per 100k) is uncorrelated with the population. More surprisingly, the nation's GDP for the year is positively correlated with the total number of suicides, implying that richer nations have higher suicide numbers. This is further confirmed as the `HDI for year` is positively correlated with the suicide rate. The `year` has a modest negative correlation with the suicide rate, suggesting that the suicide rate has declined over time.

Video Transcoding Time Dataset: Figure 3 shows the heatmap of Pearson correlation matrix for the video transcoding time dataset columns. The `b_size` feature is invalid as all instances were 0 in the dataset. The `o_width` and `o_height` have the highest absolute correlation with the target variables. The large positive correlation implies that higher output video resolutions leads to larger transcoding times.

Question 2: Plot the histogram of numerical features.

If the distribution of a feature has high skewness, the feature can be standardized by shifting and scaling the feature to have zero mean and unit variance.

Bike Sharing Dataset

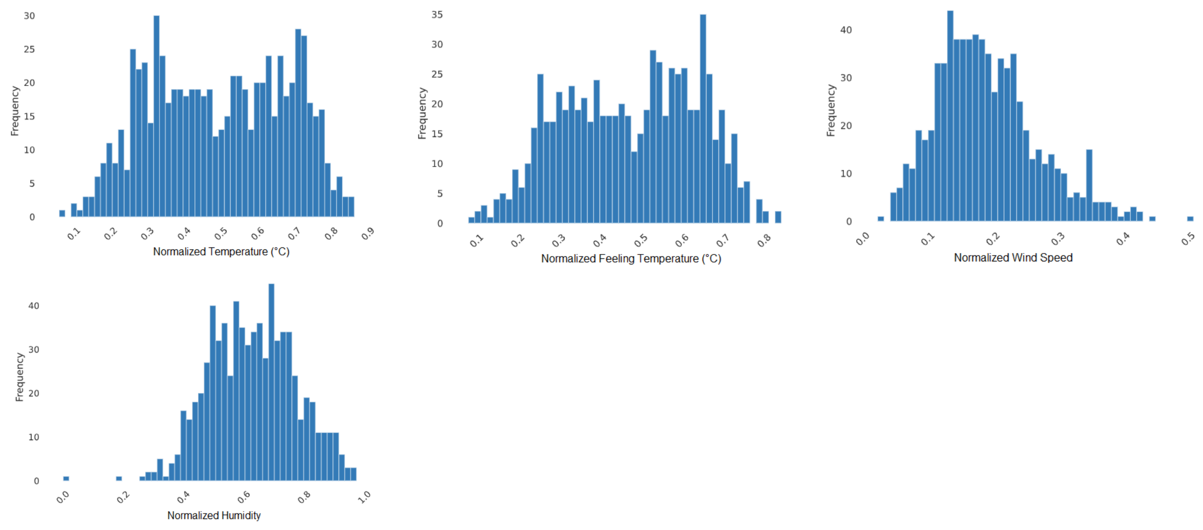


Figure 4: Histogram of numerical features for bike sharing dataset.

Suicide Rates Overview Dataset

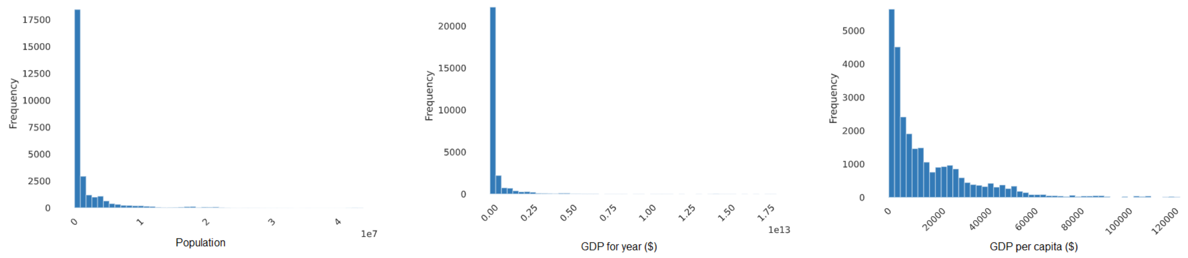


Figure 5: Histogram of numerical features for suicide rates overview dataset.

Video Transcoding Time Dataset

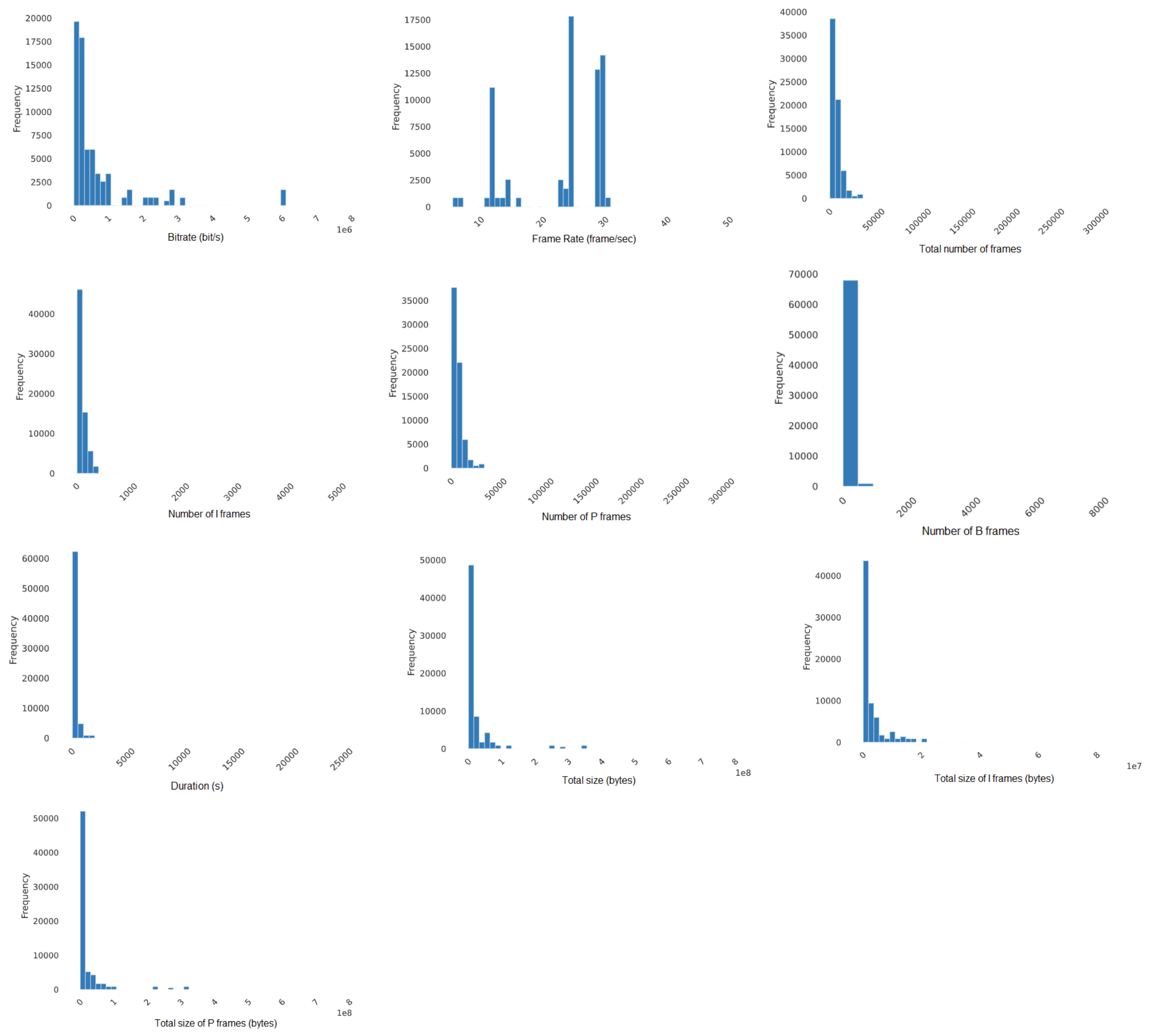


Figure 6: Histogram of numerical features for video transcoding time dataset.

Question 3: Inspect box plot of categorical features vs target variable.

Bike Sharing Dataset

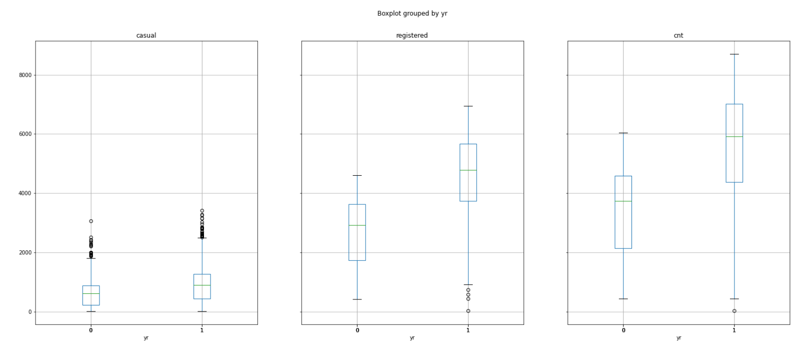


Figure 7: Box plot of casual (left), registered (middle), and total (right) bike users categorized by year.

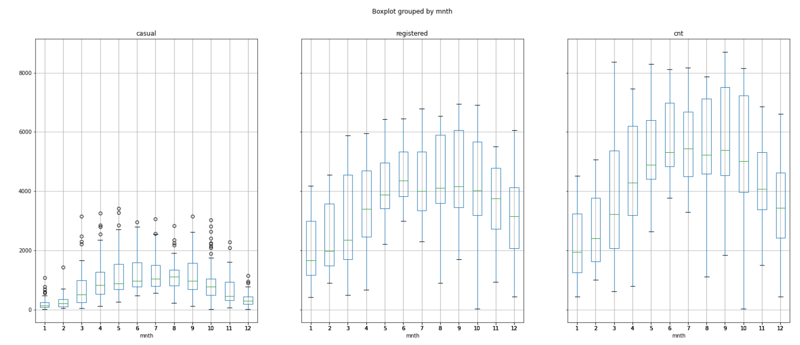


Figure 8: Box plot of casual (left), registered (middle), and total (right) bike users categorized by month.

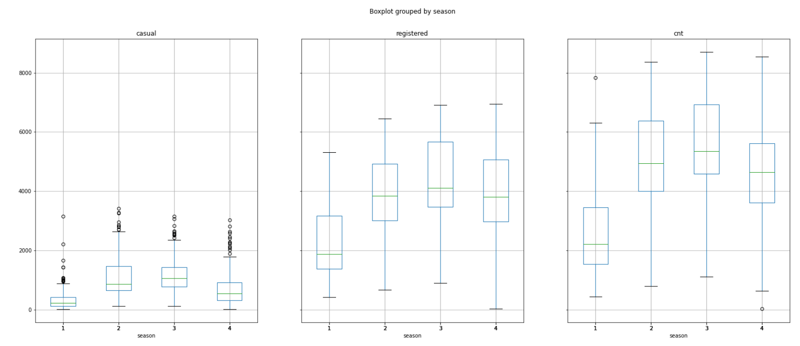


Figure 9: Box plot of casual (left), registered (middle), and total (right) bike users categorized by season.

The box plots of the casual, registered and total bike users categorized by the categorical features of the bike sharing dataset are given in Figures 7-13. From Figure 7, we can see that the average number of bike rentals increased from 2011 to 2012. Figure 8 and 9 highlights the annual trend of all bike rentals– namely being low during Winter, gradually rising through Spring, reaching a peak during Summer, and slowly falling back down during Fall. Figure 10 highlights that all bike rentals decrease as the weather becomes more adverse. Finally, Figures 11, 12, and 13 highlight the differences between casual and registered bike rental users– namely that casual bike users rent during holidays and weekends, whereas registered bike users rent during working (week)days.

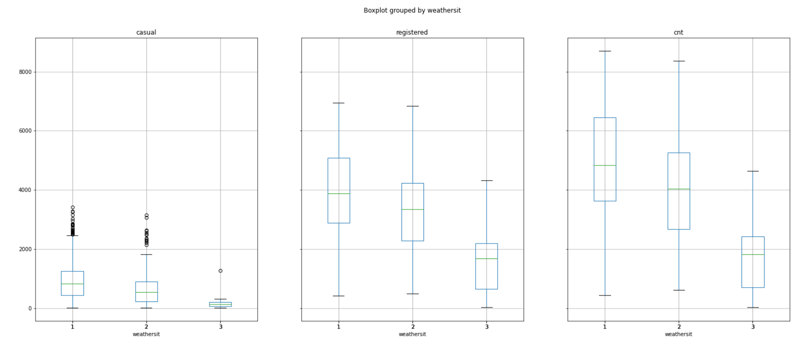


Figure 10: Box plot of casual (left), registered (middle), and total (right) bike users categorized by weather. Weather labels are as follows: 1 = Clear, Few clouds, Partly cloudy, Partly cloudy; 2 = Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist; 3 = Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds; 4 = Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog.

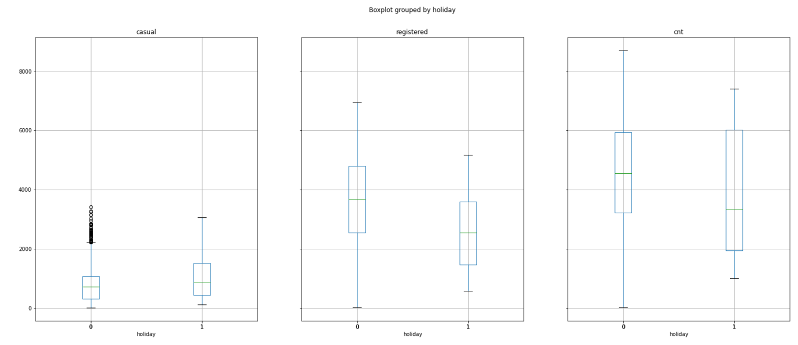


Figure 11: Box plot of casual (left), registered (middle), and total (right) bike users categorized by holiday.

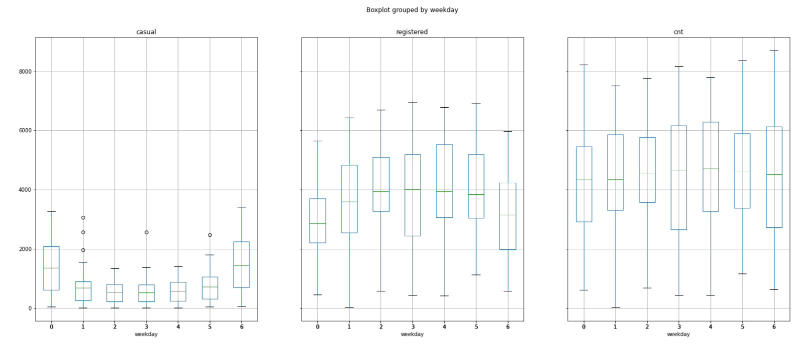


Figure 12: Box plot of casual (left), registered (middle), and total (right) bike users categorized by day of the week, starting Monday.

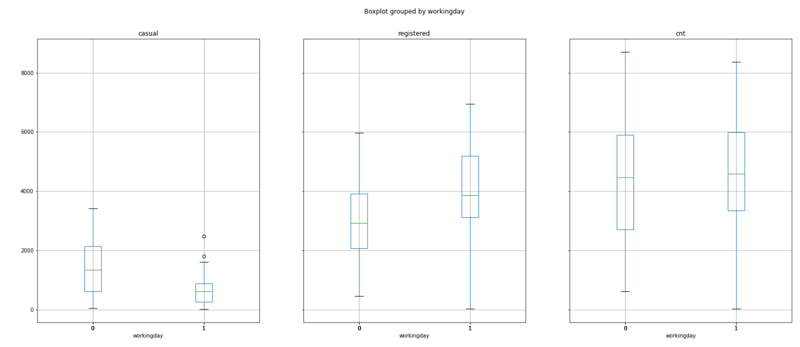


Figure 13: Box plot of casual (left), registered (middle), and total (right) bike users categorized by if it is a working day. If the day is neither a weekend nor a holiday the label is 1, otherwise it is 0.

Suicide Rates Overview Dataset

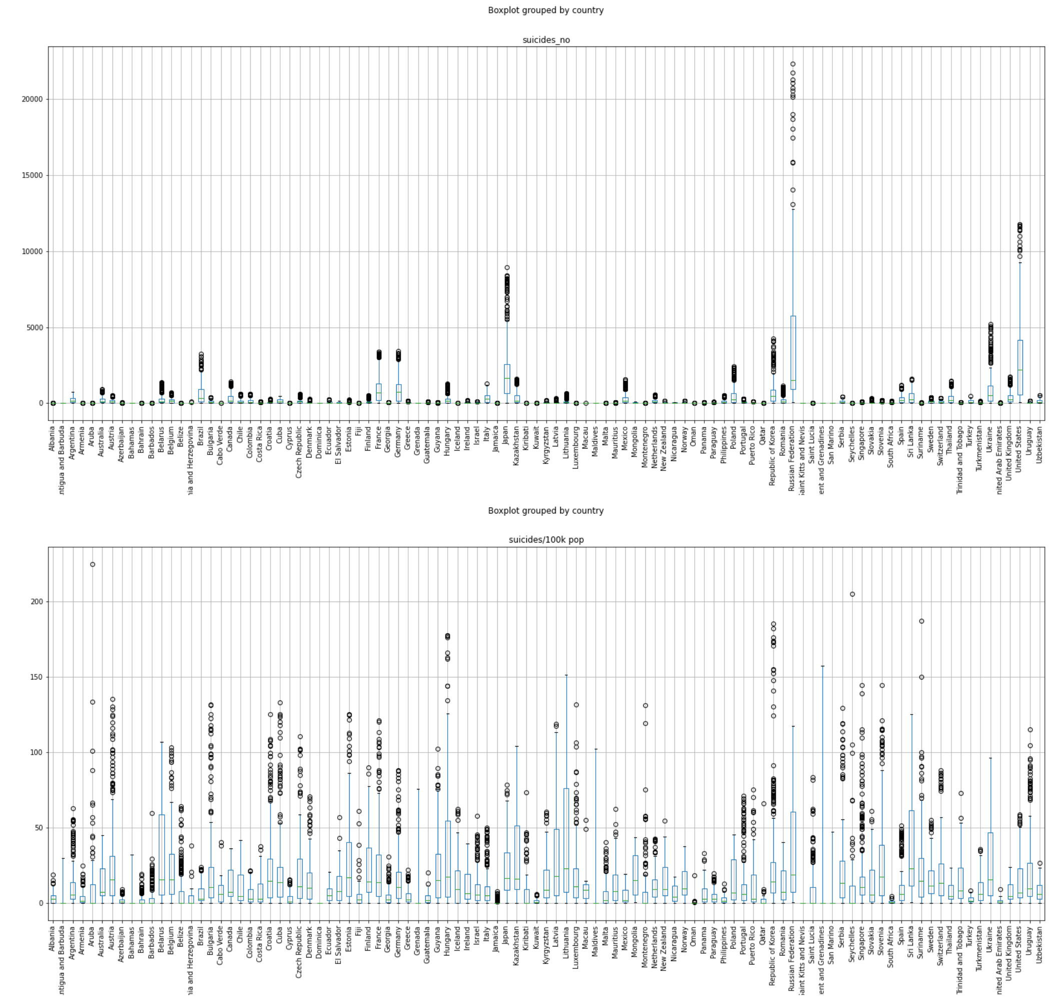
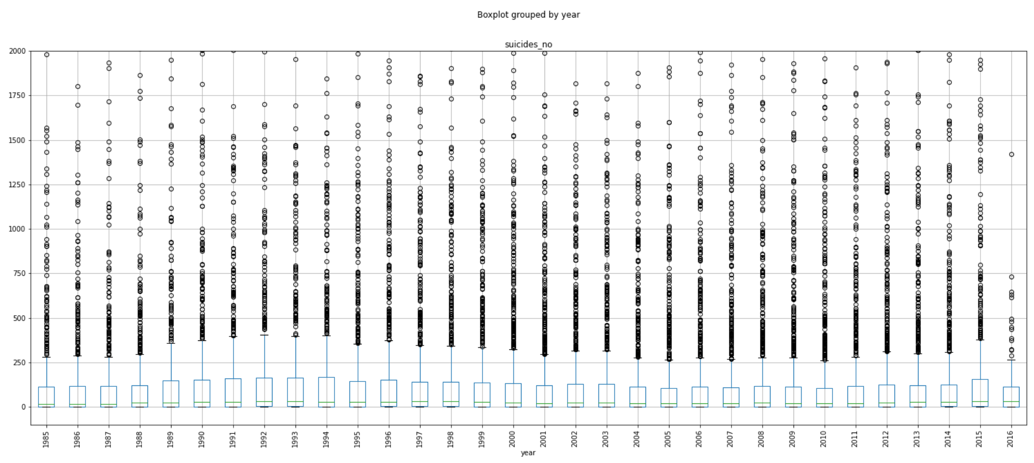


Figure 14: Box plot of number of suicides (top), and suicide rate (bottom) categorized by country.

The box plots of the total number of suicides and suicide rate (per 100k) categorized by the categorical features of the suicide rate overview dataset are given in Figures 14-18. From all the figures we can see there is many outlier points (seen as the black markers). Figure 15 shows that the total number of suicides and suicide rate were largest in the 1990s, and gradually decreased into the new millennium. However, there is a small uptick in both total suicides and suicide rate in the last two years of 2015 and 2016. Figure 16 highlights that both the total number of suicides and suicide rate is highest among the male sex. Figures 17 and 18 indicate that the suicide rate is highest amongst the oldest members of society, corresponding with those who are older that 75 years and are from the G.I generation.



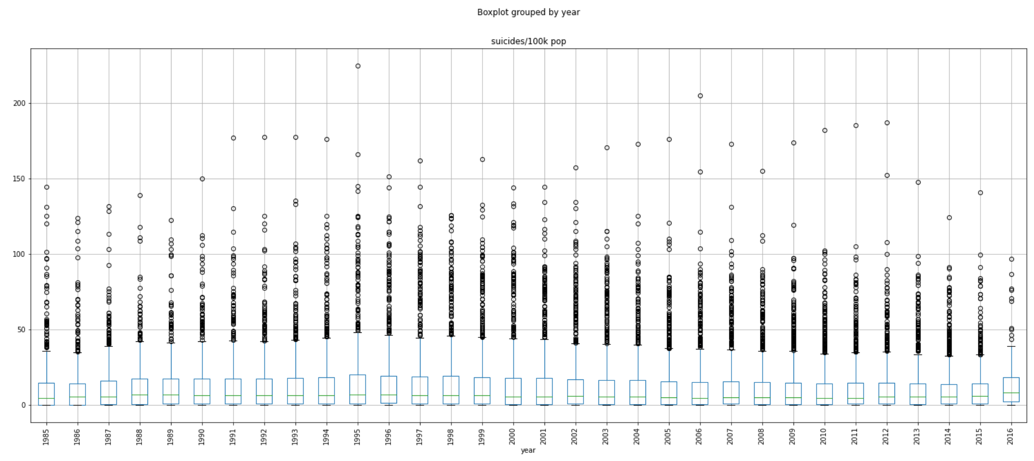
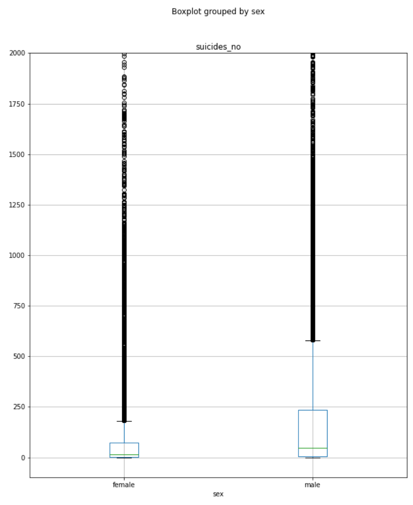


Figure 15: Box plot of number of suicides (top), and suicide rate (bottom) categorized by year.



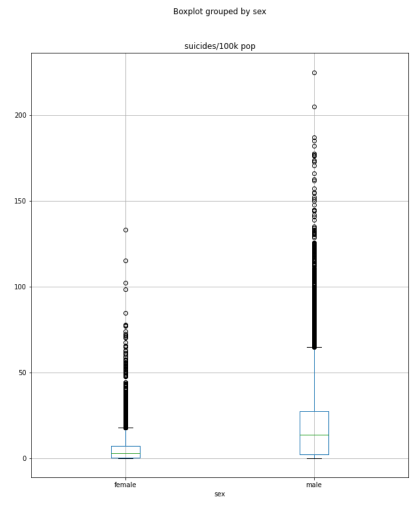
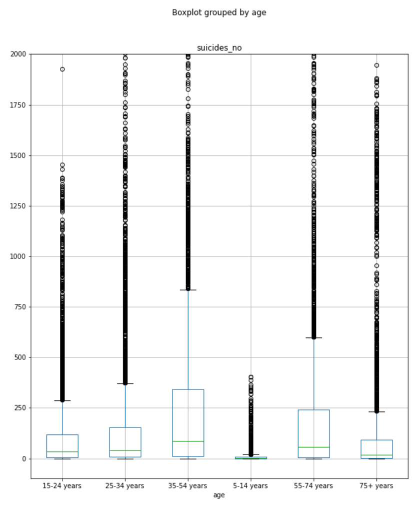


Figure 16: Box plot of number of suicides (left), and suicide rate (right) categorized by sex.



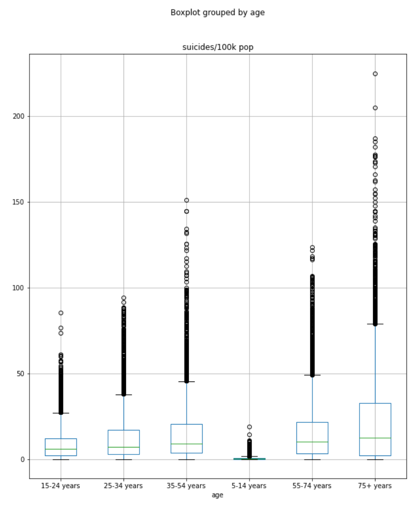
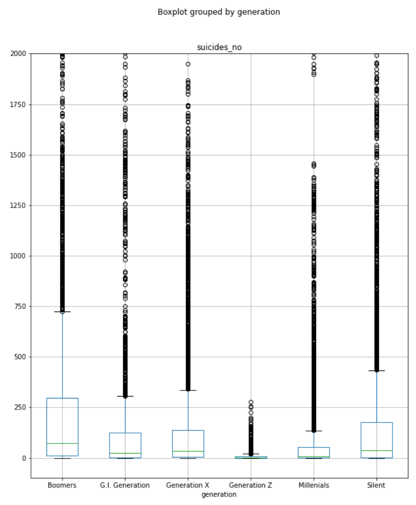


Figure 17: Box plot of number of suicides (left), and suicide rate (right) categorized by age group.



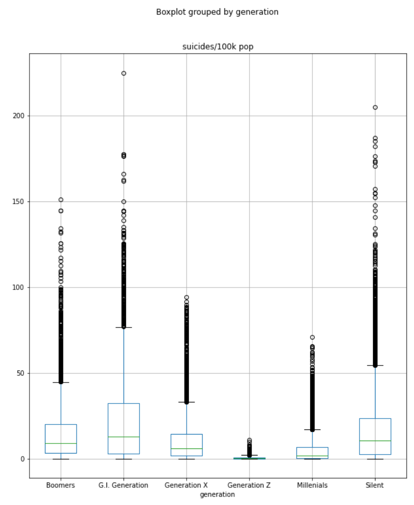


Figure 18: Box plot of number of suicides (left), and suicide rate (right) categorized by generation.

Video Transcoding Time Dataset

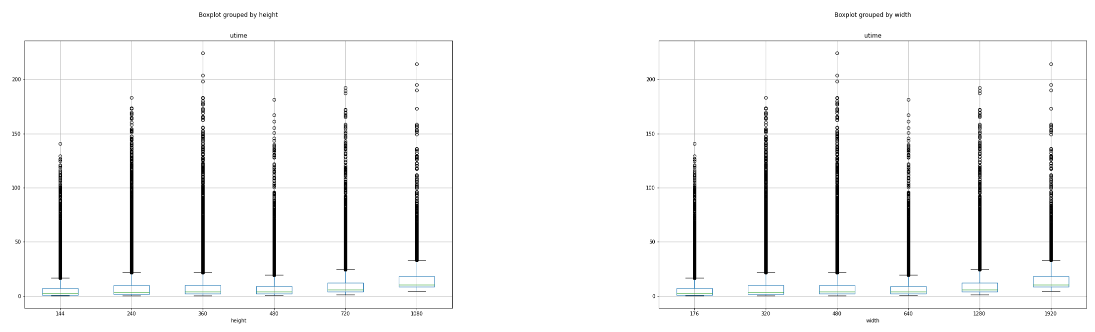


Figure 19: Box plot of total transcoding time categorized by video height (left) and width (right).

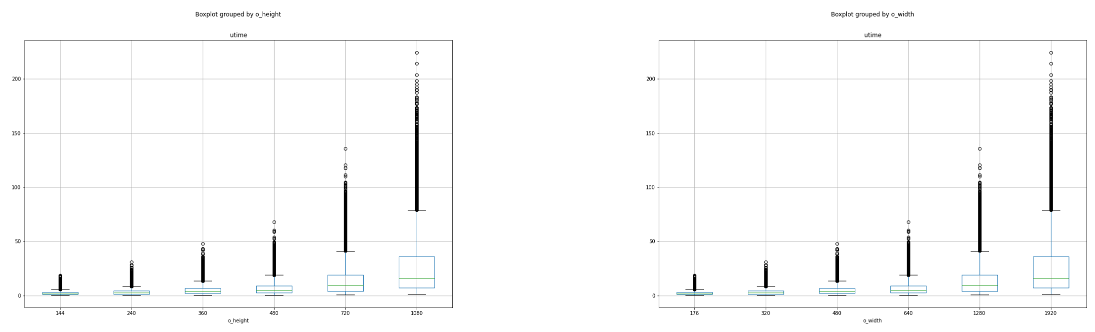


Figure 20: Box plot of total transcoding time categorized by output video height (left) and width (right).

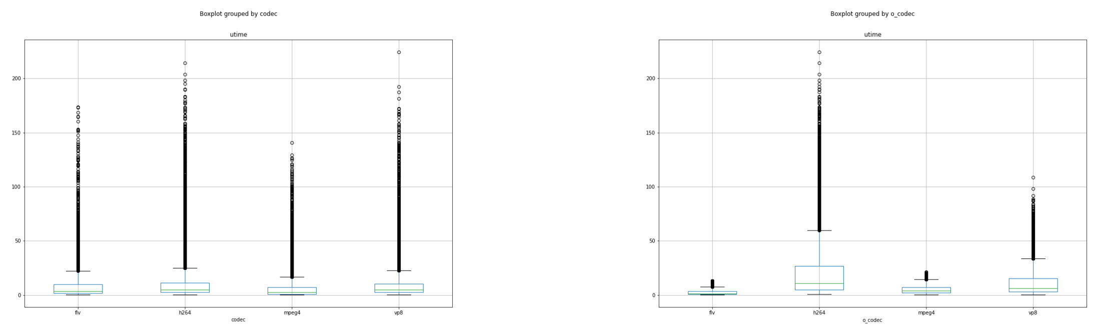
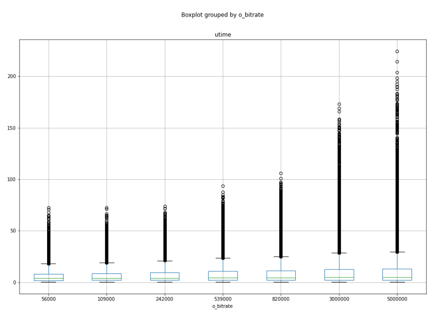


Figure 21: Box plot of total transcoding time categorized by input video (left) and output video (right) codec.



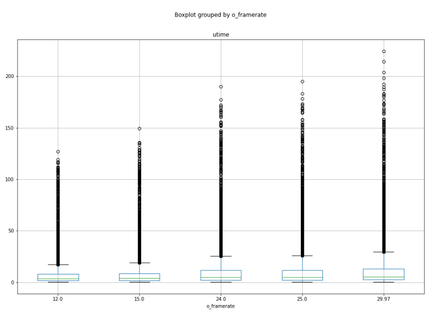


Figure 22: Box plot of total transcoding time categorized by output video bitrate.

Figure 23: Box plot of total transcoding time categorized by output video frame rate.

The box plots of the video transcoding time categorized by the categorical features of the video transcoding time dataset are given in Figures 19-23. Figures 19 and 20 highlight that the transcoding time increases as the video resolution increases, with a more clear monotonic increase trend observed given the output video resolution. Figure 21 highlights that the transcoding time is largest for videos using the `h264` codec, followed by videos using the `vp8` codec. Both Figure 22 and 23 highlight that the video transcoding time increases as the output bitrate and framerate increase.

Question 4: For bike sharing dataset, plot the count number per day for a few months.

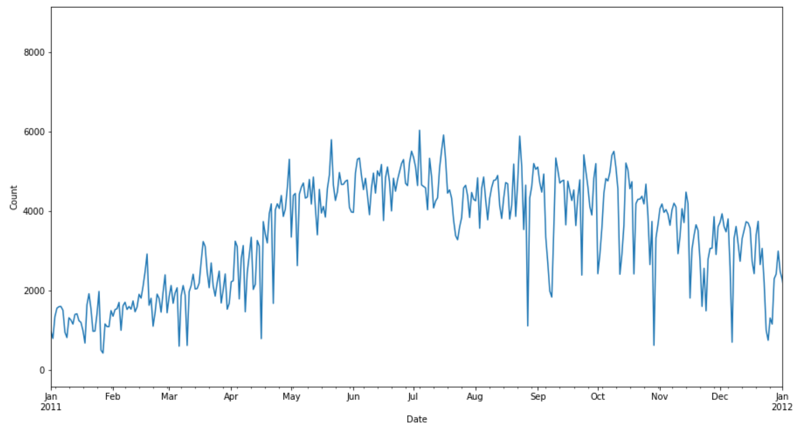


Figure 24: Total bike sharing users per day for the year 2011.

The total count number of bike rentals per day for the year 2011 is plotted in Figure 24. It is difficult to identity repeating patterns in every month, however, it appears there is a sharp drop in the total count approximately every week or two weeks. This could be because of the drop in registered bike users over the weekend. More generally, a trend can be seen of the total number of bike users rising over the course of the year, reaching a peak in the Summer, and then gradually decreasing into the Fall.

Question 5: For the suicide rate dataset, pick the top 10 countries that have the longest timespan of records (in terms of years). Plot the suicide rate against time for different age groups and gender.

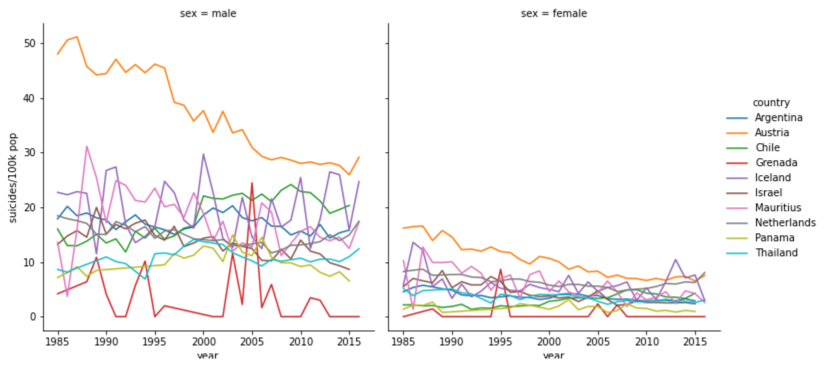


Figure 25: Suicide rate over time for different genders. The 10 countries shown had the longest timespan of records.

The top 10 countries that have the longest timespan of records (in terms of years) within the suicide rates overview dataset are selected: Argentina, Austria, Chile, Grenada, Iceland, Israel, Mauritius, Netherlands, Panama, and Thailand. These countries have records for the span of 30-31 years. Figure 25 and 26 plot the suicide rate for these countries overtime, categorized by different genders and age groups respectively. From both figures it can be seen that the suicide rate has gradually fallen over the span of 30 years. From Figure 25, it is clear that the suicide rate is higher amongst males than females. From Figure 26, it can be suggested that the suicide rate increases with age.

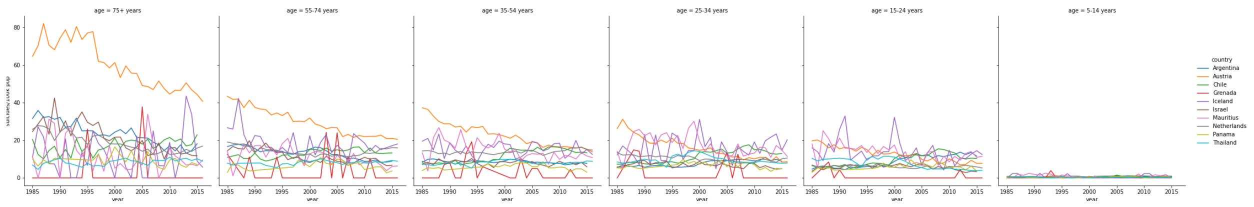


Figure 26: Suicide rate over time for different age groups. The 10 countries shown had the longest timespan of records.

Question 6: For video transcoding time dataset, plot the distribution of video transcoding times.

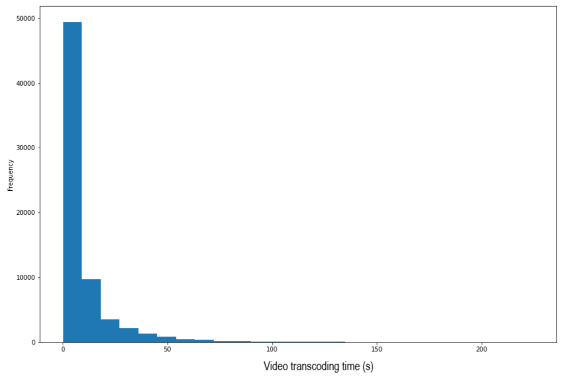


Figure 27: Histogram of video transcoding times.

The distribution of video transcoding times is given in Figure 27. It can be observed that there is a high skewness to the transcoding time, with the vast majority of transcoding times falling under 10 seconds. The mean and median transcoding times are given as follows:

- Mean transcoding time: 9.996 seconds
- Median transcoding time: 4.408 seconds

Handling Categorical Features

In order to handle categorical features, the `OrdinalEncoder` and `OneHotEncoder` transformers from the `sklearn.preprocessing` library are used. The `OrdinalEncoder` transformer is used to categorize features by assigning a scalar. For features that are "one-hot" encoded, a new column is made per category of the feature. If the feature is binary, we drop the first column as it is redundant. When mapping feature cateogires to a scalar, it is important to preserve the order. Therefore, for each dataset, we ensure to define a mapping dictionary that can be used to transform the relevant features. Below, we define the features of each dataset that use an ordinal encoding and one-hot encoding:

Bike Sharing Dataset:

- *Ordinal encoding*: `yr, mnth, weekday, weathersit, season`. These features already had pre-defined scalar mappings (see original dataset description for details).
- *One-hot encoding*: `holiday, workingday`. These features are binary, and so no additional columns are added to the dataset.

Suicide Rates Overview Dataset:

- *Ordinal encoding*: `age, generation`. These features are mapped such that the youngest age group corresponded to smallest scalar, and the oldest generation corresponded to the smallest scalar.
- *One-hot encoding*: `sex, country`. The sex feature is binary (male or female), so only one column is used for this feature. The country feature is mapped to continental regions using the library `country_converter`. Countries belonging to Asia or America are mapped to sub-continental regions as defined by the *Statistics Division of the United Nations Secretariat*. All other countries were mapped to their respective continents. In total, the country feature spans 12 columns.

Video Transcoding Time Dataset:

- *Ordinal encoding*: `width, height, o_width, o_height`. These features are mapped such that a larger width or height has a larger scalar value.
- *One-hot encoding*: `codec, o_codec`. There are 4 codec standards used, and therefore each feature has 4 respective columns.

Question 7: Explain the trade-off between one-hot encoding and scalar encoding of categorical variables.

One-hot encoding discards information relating categories within a feature, such as order. Therefore, when performing regression using, for example, a linear model, the model treats each category of a feature as if they were independent features. Ordinal (scalar) encoding preserves category order, which can enable the linear model to better learn how the categorical feature is correlated with the target variable.

Ordinal encoding, however, assumes there is some continuous and monotonic approximation between feature categories. This is not necessarily true for some time categories such as days and months, since these features are cyclical. Such features may not benefit from ordinal encoding as the cyclically nature of the feature is lost i.e. if Monday is encoded as a 1 and Sunday as a 7, Monday and Sunday are no longer days close to each other.

Therefore, the trade-off between one-hot and ordinal encoding is whether we would like to preserve order by enforcing monotonicity, or whether we would like to completely lose information relating feature categories to each other by enforcing independency.

Standardization

Question 8: Standardize feature columns and prepare them for training.

In order to standarize numerical features to have zero mean and unit variance, we utilize the `StandardScaler` transformer from the `sklearn.preprocessing` library. Below, we define the features of each dataset that are numerical and appropriately standardized:

Bike Sharing Dataset: `temp, atemp, hum, windspeed`

Suicide Rates Overview Dataset: `population, gdp_for_year ($), gdp_per_capita ($)`. Note that the `HDI for year` feature was dropped.

Video Transcoding Time Dataset: `duration, bitrate, frames, framerate, i, p, b, i_size, p_size, size, o_bitrate, o_framerate`

Feature Selection

Question 9: Select the most important features using mutual information and F score.

Selecting the most important features is a vital step as it prevents the model from trying to fit uncorrelated or potentially noisy features. If the most important features are correctly selected, we can expect the test RMSE to improve.

For each dataset, we utilize the `mutual_info_regression` and `f_regression` functions from the `sklearn.feature_selection` library to score features in terms of importance for predicting the target variable. The scores for numerical and ordinal features are given below. One-hot encoded features that are not binary are not included for brevity since there are several scores associated for each category type. Scatter plots of features versus target variables for 3 most important features ranked by their mutual information are also shown for each dataset.

Bike Sharing Dataset

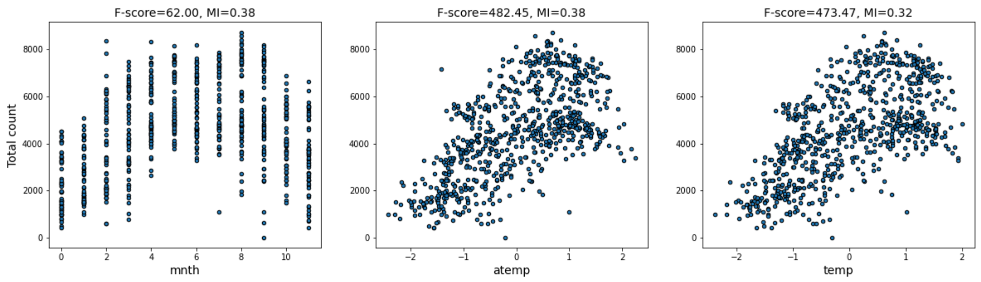


Figure 28: Scatter plot of features versus total count for top 3 features in bike sharing dataset ranked by their mutual information.

Feature	F-score	MI
temp	473.47171053497703	0.32416413952289336
atemp	482.4543105289903	0.37577424435118
hum	7.46193999634535	0.10356453310857372
windspeed	42.437841593463475	0.05947348255770457
yr	344.8905855356845	0.2751954510237773
mnth	62.004624548332224	0.3799891859760054
weekday	3.331091365174596	0.044174198875547965
weathersit	70.7292978292076	0.06475718259311058
season	143.9676525909185	0.21543146599433083
holiday	3.421441039972207	0.011528751454825459
workingday	2.7367422831912496	0.022861694127679133

Table 1: Feature scores for the bike sharing dataset.

Suicide Rates Overview Dataset

Feature	F-score	MI
population	1.909580251300788	0.3399769969418598
gdp_for_year ($)	17.732464197730646	0.3439769398291226
gdp_per_capita ($)	0.08864797749177898	0.33157245600938623
year	42.45575553449209	0.0
age	4210.793838223362	0.2715846432407778
generation	3530.554883117733	0.15317267277619884
sex	5035.427899106153	0.13649599261814682

Table 2: Feature scores for the suicide rates overview dataset. The country feature has been omitted from the table for brevity.

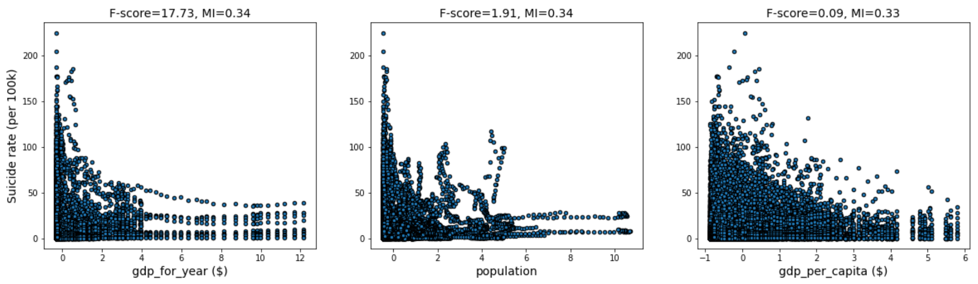


Figure 29: Scatter plot of features versus suicide rate for top 3 features in suicide rates overview dataset ranked by their mutual information.

Video Transcoding Time Dataset

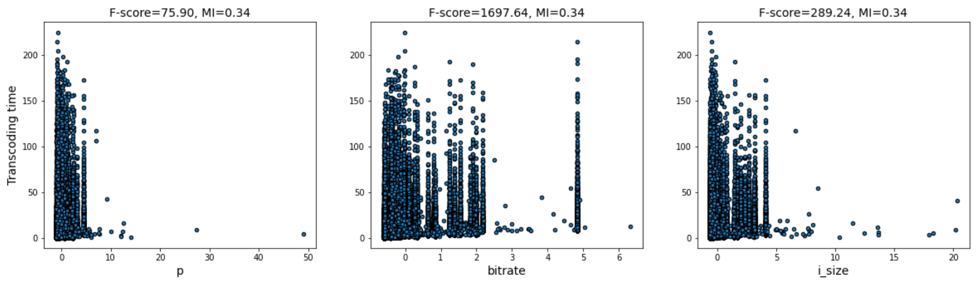


Figure 30: Scatter plot of features versus transcoding time for top 3 features in video transcoding dataset ranked by their mutual information.

Feature	F-score	MI
duration	2.1053989872457146	0.341375729168619
bitrate	1697.6446721194739	0.34169659186456336
frames	75.50745421950923	0.34097389395671396
framerate	435.6708959426289	0.19399801614058898
i	23.521143697015813	0.305968811328877
p	75.90071258286414	0.3417290161968243
b	1.8169109847941567	0.0025249848614798953
i_size	289.241036227849	0.34156672252406306
p_size	662.1102054976559	0.34156672252406306
size	654.6205699213642	0.34156672252406306
o_bitrate	1703.902785174879	0.023323707380182412
o_framerate	752.7024732737266	0.012804993535463272
width	927.6365410418391	0.14600514549582488
height	927.6365410418391	0.14940095634377926
o_width	20207.14880337625	0.3140940089730817
o_height	20207.14880337625	0.312298404944777

Table 3: Feature scores for the video transcoding time dataset. The codec and o_codec feature has been omitted from the table for brevity.

Linear Regression

Question 10: Explain how each regularization scheme affects the learned hypotheses.

The objective functions for Ordinary Least Squares (OLS), Lasso and Ridge can be formulated respectively as:

Let y be the target variable, x represent the features, w represent the weights assigned to each feature and λ represent regularization strength.

Ordinary Least Squares:

$$J(W) = \frac{1}{2N}\sum_{i=1}^{M}(y_{\mathrm{i}} - \sum_{j=0}^{P} w_{\mathrm{j}}x_{\mathrm{ij}})^2$$

Lasso Regression:

$$J(W) = \frac{1}{2N}\sum_{i=1}^{M}(y_{\mathrm{i}} - \sum_{j=0}^{P} w_{\mathrm{j}}x_{\mathrm{ij}})^2 + \lambda\sum_{j=o}^{P}|w_{\mathrm{j}}|$$

Ridge Regression:

$$J(W) = \frac{1}{2N}\sum_{i=1}^{M}(y_{\mathrm{i}} - \sum_{j=0}^{P} w_{\mathrm{j}}x_{\mathrm{ij}})^2 + \lambda\sum_{j=o}^{P}{w_{\mathrm{j}}}^2$$

$$OLS : \min_{\beta} ||Y - X\beta||_2{}^2$$

$$Lasso : \min_{\beta} ||Y - X\beta||_2{}^2 + \lambda||\beta||_1$$

$$Ridge : \min_{\beta} ||Y - X\beta||_2{}^2 + \lambda||\beta||_2$$

Regularization encourages generalization by yielding simpler models to prevent overfitting. It attempts to improve generalizability by increasing estimator bias while reducing variance, causing many of the weights to be very small, leading to a simpler model which performs well on unseen test data but may have lower overall RMSE. For both kinds of regularization, as regularization strength increases, more weights in the learned model are set to 0. This is due to the fact that the L1 and L2 regularization are formulated as minimizing the weights. Thus, to prevent the regularization portion of the least squares equation from becoming too large, the optimization program will attempt to minimize the weights, ideally the weights should approach 0. However, as models become more and more simpler, they start losing the most important weights required promoting the salient features to contribute to the final estimation. As a result, very large values of regularization strength will cause the test accuracy to drop dramatically (underfitting). However, finite values of regularization strength can help make the model more generalizable on unseen test data, preventing overfitting. Regularization also makes the model more stable by reducing variance and sensitivity to outliers and increasing bias.

For L_1 regularization, the learned hypotheses is that only a fraction of features are active in the linear model. Hence, the L_1 regularization term is added for screening purpose. While for L_2 regularization, the learned hypotheses is that all features are active but overfitting the training data. Hence, L_2 regularization term serves for shrinkage purpose.

L_1 regularization is suitable for feature selection and construction of simple and sparse linear regression models, where features associated with 0 weights can be discarded. This is because L_1 regularization encourages all kinds of weights to shrink to 0, regardless of size of w. L_1 regularization is more likely to zero out coefficients than L_2 regularization for similar test accuracies because it assumes priors on the weights sampled from an isotropic Laplace distribution (linear descent), which has a much lower Q factor than Gaussian distribution (exponential descent). Thus, only a sub-selection of features are active in the learned hypothesis.

L_1 regularization manage to sparsify the coefficients we are going to learn. After applying this regularization, most of the coefficient will be restricted to zero. This regularization is always applied to huge number of features training data so that we can use this regularization to remove some feature altogether, namely this works well for feature selection.This is why the performance of the model will deteriorate a little bit.

L_2 regularization, which assumes priors on the weights sampled from a unit Gaussian distribution, is suitable for reducing the effects of collinear features, which can lead to increased variance of the model. This is because the subgradient of $||w||^2$ depends on not just the sign of w but also the magnitude of w, which can be thought of as scaling the variance of weights, reducing dependence of the model on few features and encouraging distributed contribution of all the features. This leads to regression models with diffuse weights compared to sparse weights from L_1 regularization. Thus, ridge regularization promotions participation of all the features in the learned hypothesis to prevent overfitting.

L_2 regularization manage to avoid overfitting on the training set and will prefer learned weight with small norm. If regularization strength selected properly, the generalization ability of the model can be improved with the introduction of regularization as we can see through the comparison between vanilla logistic regression without regularization and model with L_2 regularization. We always introduce strong L_2 regularization to the complex model to avoid the model just repeat the pattern shown in training set even though such pattern may be contaminated.

Question 11: Report your choice of the best regularization scheme along with the optimal penalty parameter and briefly explain how it can be computed.

Bike Sharing Dataset

For bike sharing dataset, Ridge regularization is the best regularization scheme on features selected using F1-regression with an optimal parameter as 10. We manually choose five most significant features based on both mutual information and F scores criteria, which are $atemp$, $temp$, yr, $season$ and $mnth$. The first two features possess the highest absolute correlation with target cnt. In order to search for the best regularization scheme systematically, we construct pipelines and perform grid search over $\{10^x | x = -3, -2, \dots, 2, 3\}$ in order to optimize the regularization term for both Lasso and Ridge regression, and report the optimal linear model that achieves the least testing RMSE among all those candidate models. The final results for bike dataset are summarized in Table 4. For simplicity, we only report the top 10 linear models with highest negative root mean squared error.

mean test score	mean train score	param_model	param_model_alpha	standardize	feature selection
-957.562714	-908.643016	Ridge	10	False	F Scores
-960.244150	-906.609171	Ridge	10	True	F Scores
-963.061829	-906.522603	Lasso	10	False	F Scores
-964.043480	-906.245211	Lasso	10	True	F Scores
-966.719134	-905.978825	Ridge	1	False	F Scores
-967.279650	-905.957106	Ridge	1	True	F Scores
-968.028970	-905.935376	Lasso	1	False	F Scores
-968.162166	-905.932865	Lasso	1	True	F Scores
-968.622795	-921.718077	Ridge	100	True	F Scores
-971.158561	-905.898555	Ridge	0.1	False	F Scores

Table 4: Top 10 Linear Regression Models with Highest Mean Test Score for Bike Sharing Dataset

Suicide Dataset

For suicide dataset, Ridge regularization is the best regularization scheme with optimal parameter as 1000. We manually choose five most significant features based on both mutual information and F scores criteria, which are sex, age, $generation$, $year$ and gdp_for_year. In order to search for the best regularization scheme systematically, we construct pipelines and perform grid search over $\{10^x | x = -3, -2, \dots, 2, 3\}$ in order to optimize the regularization term for both Lasso and Ridge regression, and report the optimal linear model that achieves the least testing RMSE among all those candidate models. The final results are summarized in Table 5. For simplicity, we only report the top 10 linear models with highest negative root mean squared error. For suicide dataset, as we can tell, both training RMSE and validation RMSE are pretty close to each other for all listed linear models with appropriate regularization.

mean test score	mean train score	param_model	param_model_alpha	standardize	feature selection
-15.288837	-15.444549	Ridge	1000	True	F Scores
-15.288837	-15.444549	Ridge	1000	True	Mutual Information
-15.291321	-15.436819	Ridge	100	False	F Scores
-15.291800	-15.437192	Lasso	0.1	True	F Scores
-15.291800	-15.437192	Lasso	0.1	True	Mutual Information
-15.292646	-15.436144	Ridge	100	True	F Scores
-15.292646	-15.436144	Ridge	100	True	Mutual Information
-15.293752	-15.436077	Lasso	0.01	False	F Scores
-15.293756	-15.436022	Ridge	10	False	F Scores
-15.293800	-15.436025	Lasso	0.01	True	F Scores
-15.293800	-15.436025	Lasso	0.01	True	Mutual Information

Table 5: Top 10 Linear Regression Models with Highest Mean Test Score for Suicide Dataset

Video Dataset

For video transcoding time dataset, Lasso regularization is the best regularization scheme on features selected using F1-regression with an optimal parameter as 0.1. We manually choose five most significant features based on both mutual information and F scores criteria and compare corresponding model performances using 10-fold cross-validation. In order to search for the best regularization scheme systematically, we construct pipelines and perform grid search over $\{10^x | x = -3, -2, \ldots, 2, 3\}$ in order to optimize the regularization term for both Lasso and Ridge regression, and report the optimal linear model that achieves the least testing RMSE among all those candidate models. The final results are summarized in Table 6. For simplicity, we only report the top 10 linear models with highest negative root mean squared error. For video transcoding time dataset, as we can tell, both training RMSE and validation RMSE are pretty close to each other for all listed linear models with appropriate regularization. Five features selected include o_width, o_height, o_codec_h264, o_codec_mpeg4 and $o\_bitrate$. The first two features possess the highest absolute correlation with the target.

mean test score	mean train score	param_model	param_model_alpha	standardize	feature selection
-11.851441	-11.856209	Lasso	0.1	True	F scores
-11.851460	-11.856118	Lasso	0.01	False	F scores
-11.851514	-11.856085	Ridge	10	False	F scores
-11.851524	-11.856085	Lasso	0.001	False	F scores
-11.851533	-11.856084	Ridge	1	False	F scores
-11.851534	-11.856084	Ridge	1	True	F scores
-11.851535	-11.856096	Ridge	10	True	F scores
-11.851535	-11.856084	Ridge	0.1	False	F scores
-11.851535	-11.856084	Ridge	0.1	True	F scores
-11.851535	-11.856084	Ridge	0.01	False	F scores

Table 6: Top 10 Linear Regression Models with Highest Mean Test Score for Video Transcoding Time Dataset

In terms of feature selection methods, $f\_regression$ achieves universally better performance compared to $mutual\_info\_regression$.

Question 12: Does feature scaling play any role (in the cases with and without regularization)?

Without Regularization

For linear models without regularization, feature scaling won't make any difference to the model, since normalizing one feature will only incur changes to the corresponding coefficient and intercept, but won't affect the fitted value of target at all. Since the data distribution is the same as before, the changes caused by feature scaling will reflect linearly on the changes in the coefficients, yielding no performance gains or losses. As a result, we see no change in test RMSE with or without feature scaling when regularization is absent. We can justify our answer by summarizing the model performances of ordinary linear regression for bike data shown in Table 7, suicide data shown in Table 8 and video data shown in Table 9 as examples:

mean test score	mean train score	param_model	standardize	feature selection
-971.893149	-905.897288	LinearRegression	False	F Scores
-971.893149	-905.897288	LinearRegression	True	F Scores
-1075.046936	-990.209271	LinearRegression	False	Mutual Information
-1075.046936	-990.209271	LinearRegression	True	Mutual Information

Table 7: Effect of Standardization on Ordinary Linear Regression for Bike Sharing Dataset

mean test score	mean train score	param_model	standardize	feature selection
-15.294124	-15.436013	LinearRegression	False	F Scores
-15.294124	-15.436013	LinearRegression	True	F Scores
-17.697450	-17.645069	LinearRegression	False	Mutual Information
-17.697450	-17.645069	LinearRegression	True	Mutual Information

Table 8: Effect of Standardization on Ordinary Linear Regression for Suicide Dataset

mean test score	mean train score	param_model	standardize	feature selection
-11.851535	-11.856084	LinearRegression	False	F scores
-11.851535	-11.856084	LinearRegression	True	F scores
-15.879762	-15.903696	LinearRegression	False	Mutual information
-15.879762	-15.903696	LinearRegression	True	Mutual information

Table 9: Effect of Standardization on Ordinary Linear Regression for Video Transcoding Time Dataset

With Regularization

While for linear models with regularization, things become a bit more complicated. Standardizing the features or not will have an influence on the regularization, since normalization will cause changes to the estimated coefficients. Therefore, same penalty term means different for raw features and scaled features. Since regularization aims at minimizing the weights of the regression model, it would remove or penalize the coeffcients of the smaller feature, while having negligible effect on the weights of the larger feature. Feature scaling ensures that the weights assigned to each feature does not get adversely affected by a 'biased regularizer penalizing smaller features' when regularization is used and leads to more stable and well-conditioned models. In addition, standardization also improves model convergence speed by making training less sensitive to scale of features. This conclusion can also be justified by looking at the model performances of both Lasso and Ridge with different regularization terms for bike data in Table 10, suicide data in Table 11 and video data in Table 12.

mean test score	mean train score	param_model	param_model_alpha	standardize	feature selection
-957.562714	-908.643016	Ridge	10	False	F Scores
-960.244150	-906.609171	Ridge	10	True	F Scores
-963.061829	-906.522603	Lasso	10	False	F Scores
-964.043480	-906.245211	Lasso	10	True	F Scores

Table 10: Effect of Standardization on Regularized Linear Regression for Bike Sharing Dataset

mean test score	mean train score	param_model	param_model_alpha	standardize	
-15.321485	-15.490397	Ridge	1000	False	F Scores
-15.288837	-15.444549	Ridge	1000	True	F Scores
-15.296255	-15.442317	Lasso	0.1	False	F Scores
-15.291800	-15.437192	Lasso	0.1	True	F Scores

Table 11: Effect of Standardization on Regularized Linear Regression for Suicide Dataset

mean test score	mean train score	param_model	param_model_alpha	standardize	feature selection
-11.851556	-11.856813	Lasso	0.1	False	F Scores
-11.851441	-11.856209	Lasso	0.1	True	F Scores
-11.851514	-11.856085	Ridge	10	False	F Scores
-11.851535	-11.856096	Ridge	10	True	F Scores

Table 12: Effect of Standardization on Regularized Linear Regression for Video Transcoding Time Dataset

Question 13: What is the meaning of p-values for different features and how can you infer the most significant features?

p-values in the linear regression model measures the probability of feature coefficients being equal to zero. Hence, if the p-value for some feature is very close to 0, we will have the confidence to say that particular feature is significant in the linear model.

A feature with a high p-value ($>$ 0.05) indicates that there is insufficient evidence to find any meaningful correlation of that feature with the changes in the target-variable. On the other hand, a feature with a low p-value ($<$ 0.05) indicates that the probability of the coefficients of that particular feature being 0 is low, which indicates that the feature is well-suited as a salient feature, contributing to changes in the target variable and ultimately rejecting the null hypothesis for that particular predictor (statistically significant feature). Thus, the most significant features will have a small p-value.

We can use python package $statsmodels$ to compute p-values for all features conveniently. For bike sharing dataset, some of the most significant features are $temp$, $atemp$, yr, $season$. For suicide dataset, some of the most significant features with very small p-value include gdp_for_year, gdp_per_capita, age and sex. For video transcoding time dataset, the most significant features with very small p-value include o_codec_h264, $o\_framerate$, $o\_bitrate$, o_codec_vp8, o_width, $bitrate$, o_codec_mpeg4.

So in summary, the p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value indicates that you can reject the null hypothesis. In other words, a predictor that has a low p-value is likely to be a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable. Conversely, a larger (insignificant) p-value suggests that changes in the predictor are not associated with changes in the response.

The $SelectFpr$ method is based on the FPR test, False Positive Rate, which means a false positive rate, which refers to the proportion of samples that we predict to be positive but actually negative, that is, the proportion of errors in a type of hypothesis test. Here, the features are filtered according to the p-value. The smaller the p-value, the better. All features with p-values lower than the threshold α that we set will be selected.

Polynomial Regression

Question 14: Look up for the most salient features and interpret them.

The most salient features are those with greatest absolute coefficients in the polynomial model. The most salient (top 5) features for the best-performing polynomial regressors for each of the three datasets are:

Bike-sharing dataset: ’temp’, ’atemp’, ’windspeed’, ’atemp∧2’, ’temp atemp’.

Suicide-rates dataset: ’sex_female’, ’sex_male’, ’sex_female∧2’, ’sex_female∧3’, ’sex_male∧2’.

Video-transcoding dataset: ’o_codec_flv’, ’o_codec_flv∧2’, ’o_codec_h264∧2’, ’o_codec_h264’, ’o_codec_mpeg4’.

For bike sharing dataset, the most salient raw features are $temp$ and $atemp$, and those two features also have the highest absolute correlation with the target. This conclusion is reasonable, since people are more willing to rent bike on warm days and avoid using bike on freezing days. Also, a higher wind speed during warm weather indicates a pleasant riding experience in the summer breeze, leading to more bike rentals. Thus, the total number of users depends largely on the temperature and wind speed.

For suicide dataset, the most salient features are various combinations of ’sex’, which had the highest absolute correlation with ‘suicides/100k pop’ target variable. It shows that men tend to commit more suicides globally than women. $population$ and gdp_for_year have high absolute correlation too which makes sense since a larger population will tend to have a larger total number of suicides. Also, the nation’s GDP for the year is positively correlated with the total number of suicides, implying that richer nations have higher suicide numbers. This is further confirmed as the $HDI\ for\ year$ is positively correlated with the suicide rate. $year$ has a modest negative correlation with the suicide rate, suggesting that the suicide rate has declined over time.

For video transcoding time dataset, the most salient raw features are combinations of output codecs. ‘o_codec’ has one of the highest absolute correlations along with ‘o_height’ and ‘o_width’. This conclusion is interpretable, since the transcoding time definitely depends on the output video size, which also depends on the output width and height in pixel used for transcoding. The output codec determines the compression and decompression time required for processing the video and the output screen size (height and width) of the video. Larger videos take more transcoding time than smaller videos.

Question 15: What degree of polynomial is best? What reasons would stop us from too much increase of the polynomial degree? How do you choose that?

Bike Dataset

For bike sharing dataset, we choose the top 5 features based on F scores and mutual information as our raw features, which is the optimal choice from above linear models. Then, we further perform grid search over polynomial degrees from 1 to 10 and report both train and validation RMSE using 10-fold cross-validation. The result is shown in Table 13 and Figure 31.

mean test score	mean train score	param_poly_transform_degree
-960.244150	-906.609171	1
-906.530720	-773.518061	2
-776.971097	-681.062692	3
-890.564468	-660.754826	4
-909.466399	-642.989202	5
-1007.217088	-631.661476	6
-1459.047616	-623.512507	7
-1713.565533	-615.879867	8
-2480.036797	-609.635291	9
-2696.825356	-602.884342	10

Table 13: Table showing performances for Polynomial Regression with Different Degrees for Bike Sharing Dataset

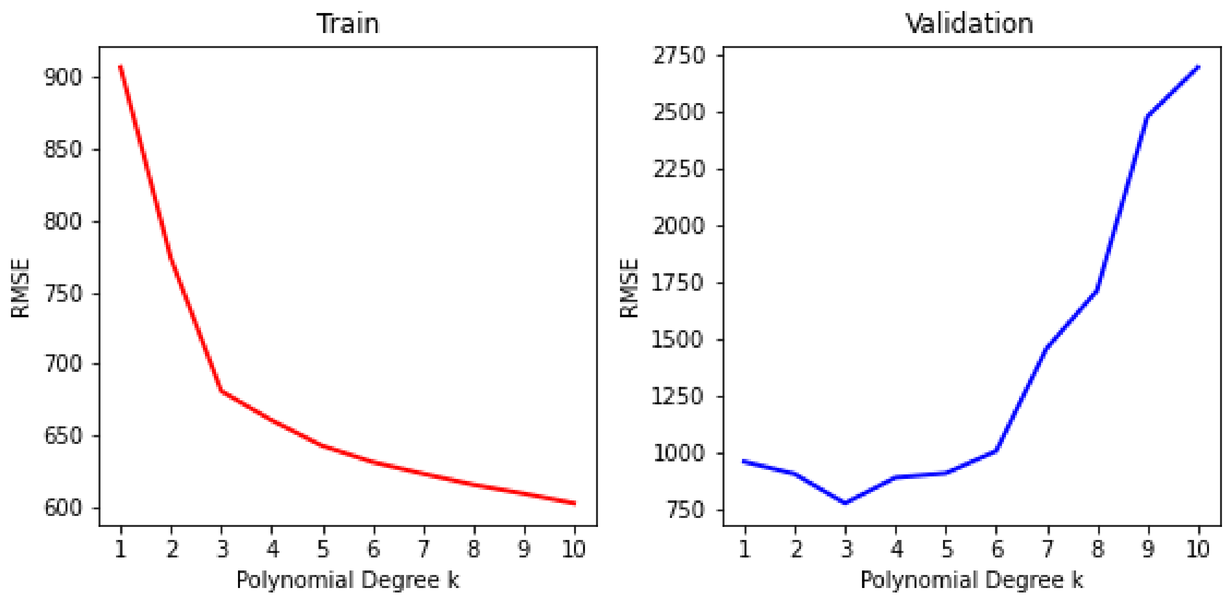


Figure 31: Figures showing performances for Polynomial Regression with Different Degrees for Bike Sharing Dataset

For bike sharing dataset, the optimal degree of polynomial we select is 3. From the validation performance we can observe that starting from polynomial degree of 3, validation RMSE has an increasing trend, which results in overfitting. Thus we should stop increasing the polynomial degree.

Suicide Dataset

For suicide dataset, we choose the top 5 features based on F scores and mutual information as our raw features, which is the optimal choice from above linear models. Then, we further perform grid search over polynomial degrees from 1 to 10 and report both train and validation RMSE using 10-fold cross-validation. The result is shown in Table 14 and Figure 32.

For suicide dataset, the optimal degree of polynomial we select is 5. Although the validation RMSE is decreasing

mean test score	mean train score	param_poly_transform_degree
-15.288837	-15.444549	1
-14.177526	-14.305988	2
-13.979272	-14.088972	3
-13.933213	-14.028516	4
-13.898875	-13.984970	5
-13.880195	-13.959508	6
-13.873712	-13.947910	7
-13.872107	-13.942448	8
-13.871541	-13.939037	9
-13.870757	-13.936194	10

Table 14: Table showing performances for Polynomial Regression with Different Degrees for Suicide Dataset

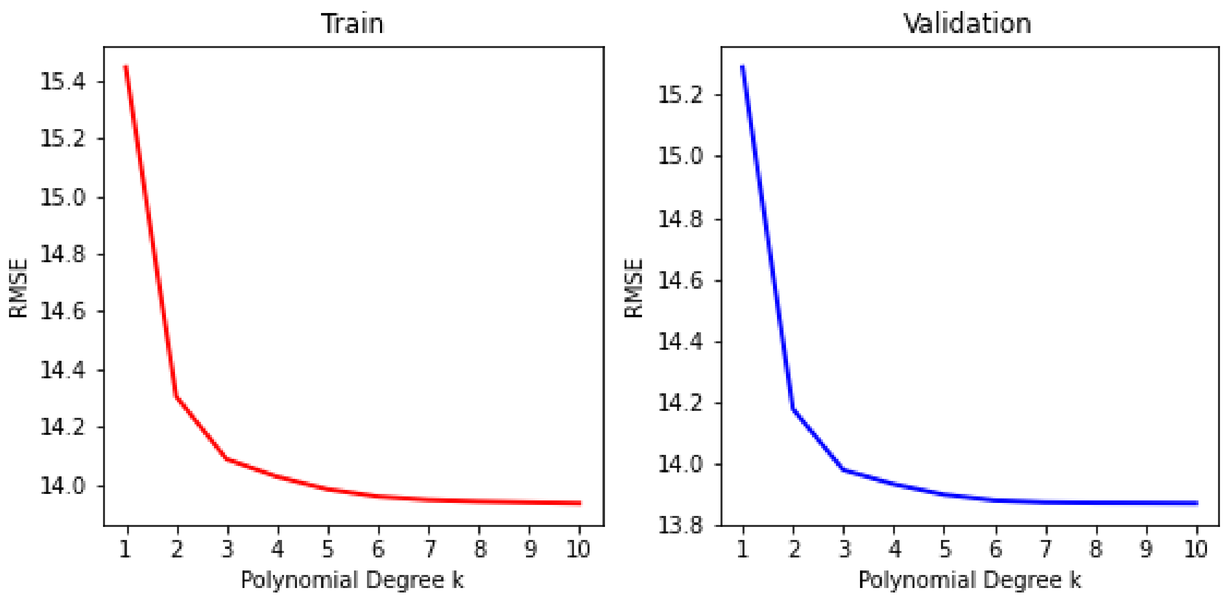


Figure 32: Figures showing performances for Polynomial Regression with Different Degrees for Suicide Dataset

monotonically, there isn't too much change after degree 5. Therefore, in order to avoid overfitting and make our polynomial model more efficient and interpretable at the same time, we choose k = 5 as the optimal degree, instead of k = 10.

Video Dataset

For video transcoding dataset, we choose the top 5 features based on F scores and mutual information as our raw features, which is the optimal choice from above linear models. Then, we further perform grid search over polynomial degrees from 1 to 10 and report both train and validation RMSE using 10-fold cross-validation. The result is shown in Table 15 and Figure 33.

For video transcoding dataset, the optimal degree of polynomial we select is 3. Although the validation RMSE is decreasing monotonically, there isn't too much change after degree 3. Therefore, in order to avoid overfitting and make our polynomial model more efficient and interpretable at the same time, we choose k = 3 as the optimal degree, instead of k = 10.

In all, there are two main reasons that prevent us from increasing the polynomial degree infinitely. First reason is from the practical consideration about computation resources. Increasing polynomial degree will result in the exponential increment of computation for the optimization. Secondly, the high degree of polynomial regression will cause the increasing model capacity, which will induce large generalization error even with

mean test score	mean train score	param_poly_transform_degree
-12.221882	-12.229764	1
-9.788216	-9.775168	2
-9.312245	-9.296554	3
-9.275881	-9.259921	4
-9.275122	-9.259120	5
-9.273424	-9.257407	6
-9.271271	-9.255239	7
-9.269442	-9.253396	8
-9.267971	-9.251914	9
-9.266833	-9.250768	10

Table 15: Table showing performances for Polynomial Regression with Different Degrees for Video Dataset

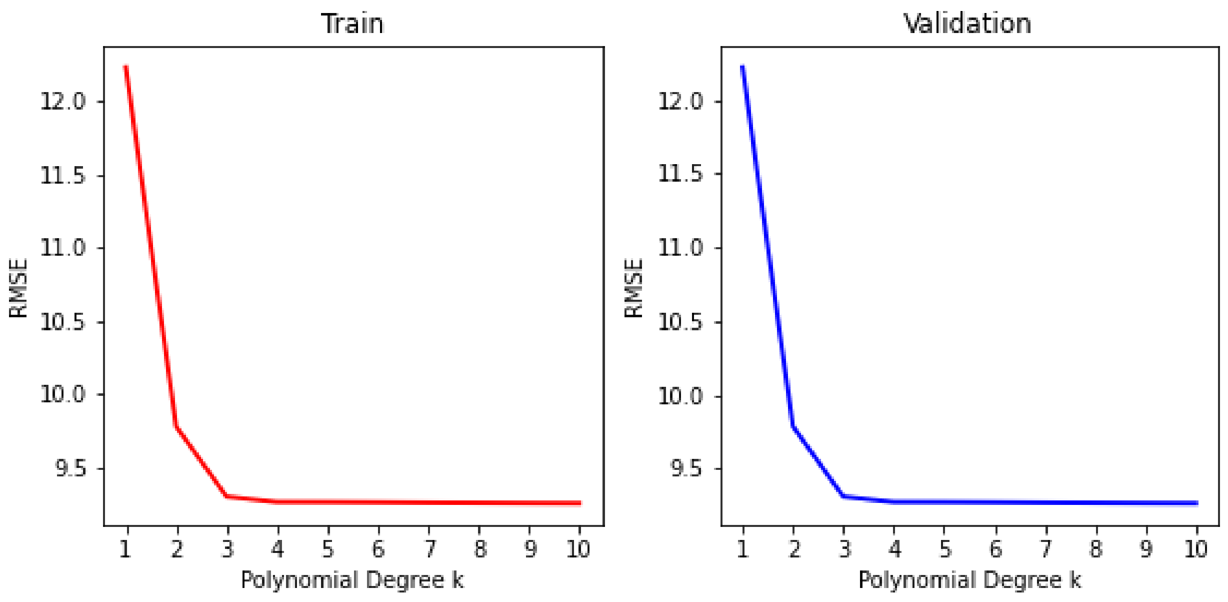


Figure 33: Figures showing performances for Polynomial Regression with Different Degrees for Video Dataset

small training error achieved.

Question 16: For the transcoding dataset, explain why it might make sense to craft inverse of certain features. Check if doing so will boost accuracy.

The reason that we need to craft such features is that the polynomial regression can never fit the inverse relationship between target and features. Based on our intuition about the trans-coding, there are several factors impacting the speed of it such as the lasting time of the whole video, size of video, bit rate of video. A reasonable crafting of new feature should be an inverse function of $\mathcal{O}_{bitrate}$ since the higher the bit-rate is, the less lasting time the whole video will have. The $\mathrm{F}-$ score of inverse bit-rate, which are much higher than the one of bit-rate, also evidence our intuitions. We are going to concatenate this feature with the original input as the input of our optimal model obtained in Question 15. The comparison between the original result and the RMSE of this method is shown as Table 16. According to the result, the crafted feature does remain after feature selection and helps the model predict the entire transcoding time precisely.

One inverse transformation we try here is $\frac{\mathcal{O}_{height} \times \mathcal{O}_{width}}{\mathcal{O}_{bitrate}}$. This new created feature is closed connected to the target, since the numerator $\mathcal{O}_{height} \times \mathcal{O}_{width}$ can be interpreted as the total number of pixels of the output video per frame for transcoding, and taking the ratio of $\mathcal{O}_{bitrate}$, which is the bits processed per unit of time for output video, we can get something proportional to the total transcoding time.

Adding this new feature to the original model, we figure out that the validation RMSE has decreased. This is a reasonable improvement to the first-order model, considering that merely one feature is added. So it will boost accuracy.

Metric	Original	Crafted Feature
RMSE	-11.852156	-11.709975

Table 16: Performance comparison between original data and crafted data for the Video Transcoding Dataset

Neural Network

Question 17: Why does the neural network do much better than linear regression?

The main reason is that neural network can capture non-linear relationship between features and target while linear regression model cannot. Furthermore, multi-layer perceptron model can fit very complicated relationship by introducing multiple hidden layers, thus we simply include all the features in this part.

From the view of model complexity, there are at least two factors that may cause the better performance of neural networks. First of all, the number of parameters of neural networks is always much larger than the one for linear regression, leading to better ability to fit the data with small generalization error if adopted proper implicit regularization. Secondly, the activation for the hidden units are non-linear functions, which has more flexibility even though the number of parameters is same as the linear model.

Dataset	Linear regression (without regularization)	Linear regression (with Lasso or Ridge regularization)	MLP
Bike	-971.893149	-957.562714	-767.629123
Suicide	-15.294124	-15.288837	-7.971124
Video	-11.851535	-11.851441	-4.576987

Table 17: Performance comparison of linear regression and MLP with most optimal hyperparameters

Question 18: Adjust your network size (number of hidden neurons and depth), and weight decay as regularization. Find a good hyper-parameter set systematically.

In order to find a good hyper-parameter setting, we follow this heuristic method: First off, we start from the neural network with only one hidden layer, and perform a grid search over the number of units in the hidden layer, ranging from 1 to 50, with weight decay penalty term alpha holding as default value 0.0001.

After find the optimal number of units for the first hidden layer, we move on to add the second hidden layer, and perform another grid search over the number of hidden units in that layer, ranging from 1 to 50, also with weight decay penalty term alpha holding as default value 0.0001.

Lastly, we fix the neural network structure, which has two hidden layers with their respective optimal number of units derived from aforementioned procedure, and grid search the regularization term over $\{10^x | x = -3, -2, \ldots, 2, 3\}$.

Bike Sharing Dataset

For bike sharing dataset, the optimal number of units for the first layer is 45, and the optimal number of units for the second layer is 48. Furthermore, the last grid search result is summarized in the Table 18.

mean test score	mean train score	param_model_alpha
-767.629123	-496.654270	1000
-781.831469	-508.209287	100
-786.528171	-520.776532	10
-788.797611	-534.398523	0.1
-789.869079	-535.165551	0.01
-790.216963	-532.572578	1
-790.583357	-536.317273	0.001

Table 18: Grid Search on Regularization Term for Bike Sharing Dataset

Hence, the optimal value of L_2 penalty parameter is 1000 for this two-layer fully-connected neural network.

Suicide Dataset

For suicide dataset, the optimal number of units for the first layer is 14, and the optimal number of units for the second layer is 42. Furthermore, the last grid search result is summarized in the Table 19.

mean test score	mean train score	param_model_alpha
-7.971124	-6.434773	100
-8.061272	-6.774465	1
-8.172961	-6.875048	0.1
-8.493228	-6.875077	0.001
-8.773230	-6.925070	0.01

Table 19: Grid Search on Regularization Term for Suicide Dataset

Hence, the optimal value of L_2 penalty parameter is 100 for this two-layer fully-connected neural network.

Video Dataset

For video transcoding time dataset, the optimal number of units for the first layer is 16, and the optimal number of units for the second layer is 40. Furthermore, the last grid search result is summarized in the Table 20.

mean test score	mean train score	param_model_alpha
-4.576987	-2.989207	10
-5.199665	-2.101557	0.1
-5.451134	-5.117978	100
-5.465738	-2.196023	1
-5.566484	-2.064144	0.01

Table 20: Grid Search on Regularization Term for Video Transcoding Time Dataset

Hence, the optimal value of L_2 penalty parameter is 10 for this two-layer fully-connected neural network.

Question 19: What activation function should be used for the output?

The activation function is none for the output, since here we are performing regression analysis. First of all, most of activation functions have limited range which restricts their application to practice. For example, the output of $tanh$ is limited to interval (0,1) while the output of $ReLu$ is limited to (0,∞). If no activation function (or a linear activation function, like identity) is used, the whole real domain is covered. Moreover, we should notice that the neural network itself can nearly approach any possible function. Even though our target does fall into the range of one specific activation function, it is still possible for neural networks to fit these data without any output activation having sufficient neurons. In all, for regression, no activation function should be adopted.

If it were classification instead of regression, then the activation function should be softmax (or sigmoid) for multi-class (or binary class) classification problems.

Question 20: What reasons would stop us from too much increase of the depth of the network?

The main reason that stops us from increasing the depth of the network too much is to avoid the overfitting problem. A deeper neural network has more trainable parameters than a shallow one, leading to a complex model with enough capacity to memorize the entire training set, consequently overfitting on the training data and generalizing poorly on the test set. Also, if a network is too deep, then the gradients will become exponentially small as the information back-propagates from the final layers towards the initial layers. This will lead to slow or premature convergence to a sub-optimal point during the training phase, with the weights in the initial layers being extremely small compared to the weights in the final layers. Moreover, it is very time-consuming and requires high compute power and memory to perform grid search to find a good hyper-parameter settings for a neural network with many hidden layers. If the network has millions of trainable weights, the neural network requires a large training set to avoid overfitting and provide robust and usable outputs during deployment.

The most important factor that prevents us from increasing the depth of the network is the trade-off between model complexity and generalization ability. Because the implicit regularization like weight decay cannot fully restrict the model capacity, using models with increasing number of parameters will result in over-fitting to the training data. Another possible reason for limited network depth can be seen in the paper of ResNet, which shows that the non-linear layer of neural networks cannot perform identical projection properly, leading to worse performance of model with more layers introduced.

Random Forest

Question 21: Model Fine Tuning

The scores for Random Forest Regression Model on each dataset are:

	Bike Dataset	Suicide Dataset	Video Dataset
Training RMSE	264.356	2.476	0.611
Testing RMSE	856.699	18.072	4.178
OOB Error	0.10765	0.118	0.008

Table 21: Random Forest Regression Model Score On Datasets

Below are the hyperparameter values chosen for each model:

	Bike Dataset	Suicide Dataset	Video Dataset
Number of Trees	50	48	20
Max Number of Features	4	8	8
Max depth	15	22	20

Table 22: Hyperparameters for each model

To find the best values for the three hyperparameters, we find maximum number of features and the number of trees first, then tune for the depth of each tree.
Take the **Bike Dataset** for example: After feature selection, we chose the eight most related features out of total of eleven. Sweep the number of trees from 1 to 200, maximum number of features from 1 to 8, and perform 10-fold cross validation. The resulting plot of average rmse and oob error to these parameters are:

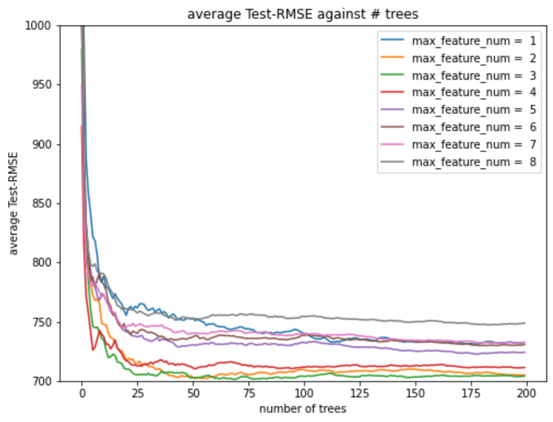


Figure 34: Average RMSE against number of trees

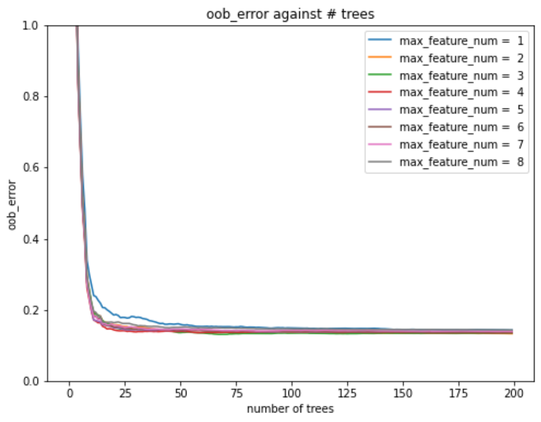


Figure 35: OOB error against number of trees

We can see from the plot, that the error scores decreases sharply when the number of trees increase from 1 to 25, then slowly decreases and tends to be stable after that. The overall performance of the model gets better with the number of trees increases, after reaching a certain point, the performance doesn't improve much. We can accordingly choose **number of trees=50**.

Observe the error score relating to the number of max features, the green line and red line that corresponds to maximum feature number of 3 and 4 are both the lowest. To decide from the two values, we can fit the regression model and see the prediction accuracy on both possibilities. Currently, the maximum depth

hasn't be decided yet, we can test around first. And from the accuracy score, when max depth is lower than 10, the scores are around 0.82 and max feature of 3 performs better. However, when max depth is over 10, the scores would fluctuate around 0.88 and max features of 4 performs better.
From there, we temporarily set max depth of 20 for now and test out the max feature combination. Fix max depth of 20 and number of trees of 50, comparing different max feature number performance: max feature number=3 achieves score of 0.8828, and max feature number of 4 achieves score of 0.8886. **maximum number of features=4** performs better.

After setting the values for number of trees and maximum number of features, we then tune for the maximum depth hyperparameter. Perform 10-fold cross validation, we got the plots:

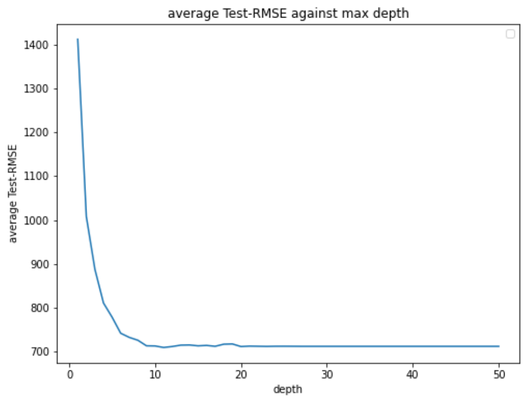


Figure 36: Average Test-RMSE against max depth

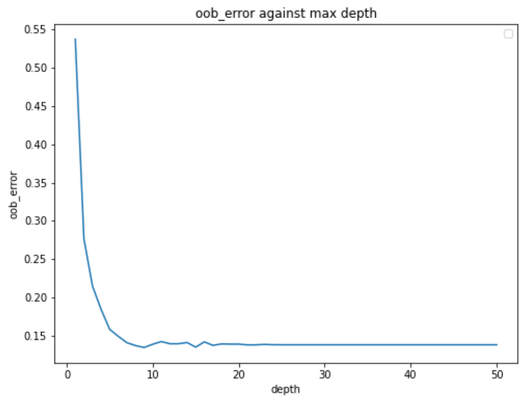


Figure 37: OOB error against max depth

It can be seen that **maximum depth=15** performs the best. Based on all the three chosen hyperparameter values, we finally get the score of the regression model: 0.89275

Do some of the hyperparameters have regularization effect?
The regularization effect is controlling the capacity of the model to prevent overfitting. Overfitting happens when the model is too flexible and adapats too much to the training data. So we need to set the hyperparameters to avoid this.
In the above three hyperparameters that we tuned, max depth and max features have regularization effect. These hyperparameters restrict the model from having too much freedom and thus reduce overfitting.

Suicide Dataset
Same approach we fine-tune model on the suicide dataset. On this dataset however, we cannot use one-hot-encoding. Observe our data after the encoder, the "country" feature was broken down to a lot of levels, which random forest would suffer from. So here, we would just apply ordinal encoding on the one hot features.
Sweep number of trees from 1 to 100, and maximum number of features from 1 to 8, while assuming the max depth to be 8 for now. After choosing these two hyperparameter values after this step, we will then see the behavior of different maximum depths from 1 to 40. After narrowing down our search space, we can furthur optimize by evaluating the model score on a smaller scope by grid search.

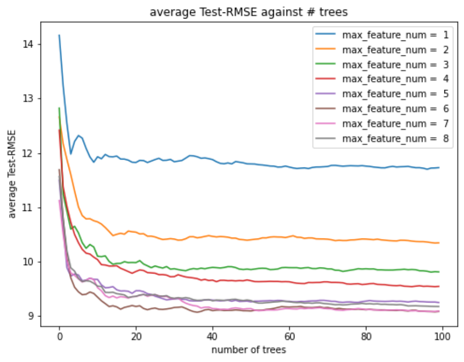


Figure 38: Average RMSE against number of trees

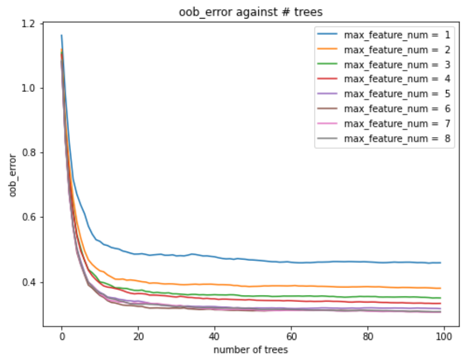


Figure 39: OOB error against number of trees

As we defined a random number for maximum depth as 8 for now, our performance isn't the best, but we can see how performance change with number of trees and maximum feature numbers. From the plots, the best number of trees is between 40 to 60, and best number for maximum feature is 8. We'll use 50 trees and 8 features to test out the maximum depth.
The maximum depth behavior as follows:

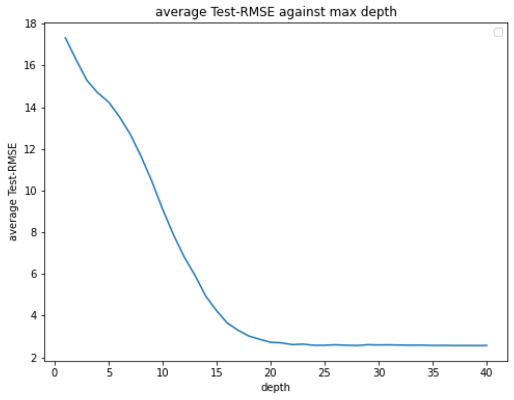


Figure 40: Average Test-RMSE against max depth

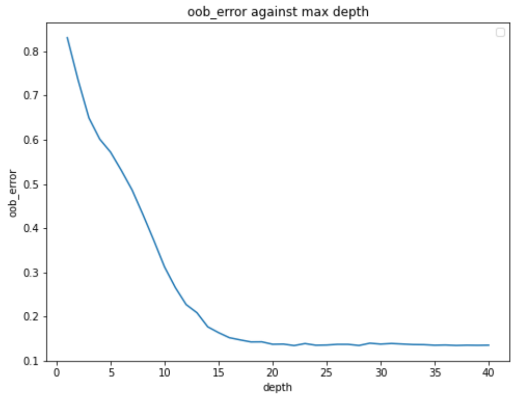


Figure 41: OOB error against max depth

We can tell from the figures that the best value for our maximum depth is around 20.
The model performance improves significantly when the max depth increases from 1 to 20, then reaches a steady state.
We have now narrowed down our scope for the hyperparameters, to further optimize our model, we can perform grid search. Our search space would be: tree number 40-50, max feature number 6-9, max depth 20-30.
The result returns **number of trees: 48; max features: 8; max depth: 22** as the best performance where the score is 0.8727 and RMSE of 2.476.

Video Dataset

For the video dataset, since our one-hot-encoding sample is very small, we can keep using the previously transformed data. But we'll compare the performance between datasets with one hot features and not. The data without one hot encoding performs better. The below hyperparameter tuning used the one with one hot encoding, our result is still accurate as the ratio of our one hot features is of low ratio in our entire sample.
Sweep number of trees from 1 to 100, and maximum number of features from 1 to 8, while temporarily set the max depth to 10. Use the different combinations on models, predict the training data to get the RMSE and OOB error. The plots are as follows:

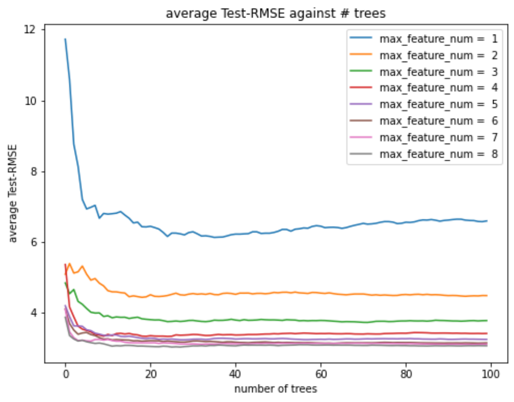


Figure 42: Average RMSE against number of trees

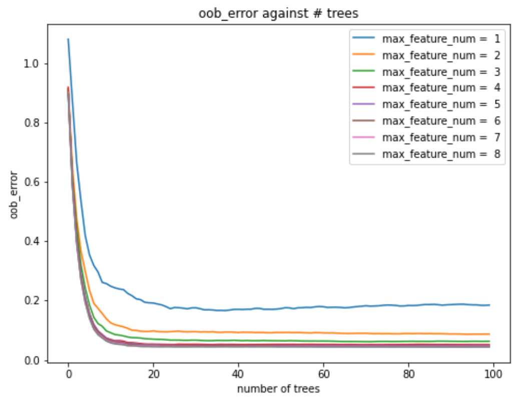


Figure 43: OOB error against number of trees

It can be seen clearly from the results that **maximum feature of 8** performs the best, as the light purple line represents the lowest error in both RMSE and OOB curve. The model performance increases significantly when the number of trees sweeps from 1 to around 15, then reaches a steady state. We can accordingly choose **number of trees=20**.

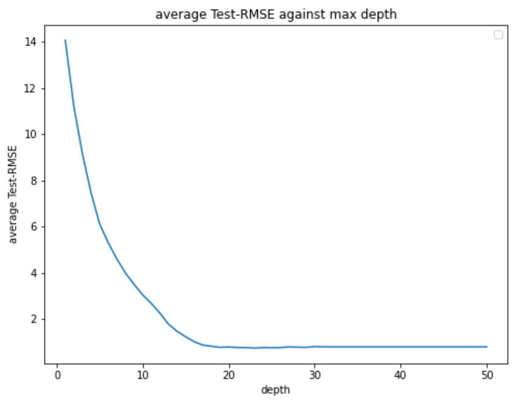


Figure 44: Average Test-RMSE against max depth

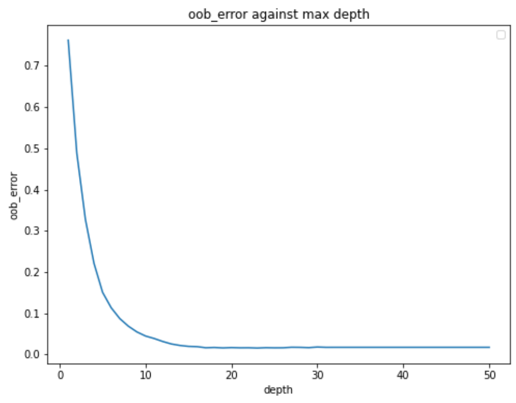


Figure 45: OOB error against max depth

We then determine the best value for max depth. Sweeping from 1 to 50, the model performance increases in a shape like 1/x curve. After reaching max depth of around 17, the performance reaches a steady state and doesn't improve much. We will choose **max depth of 20**. The combining hyperparameter set on the model achieves high performance, with model score of 0.987 and very low training RMSE (0.652) and testing RMSE (4.364) after applying 10-fold cross validation. If we used video dataset after preprocessing without one hot encoding, our score is 0.991, training rmse 0.611 and testing rmse 4.178.

Question 22: Why does random forest perform well?

The process of random forest is: 1) create bootstrap dataset with subset of variables; 2) fit decision tree; 3) repeat many times and record the predictions; 4) choose the class with more votes to use for prediction.
Random forest has high variation, which allows a high level of flexibility. Other models such as linear regression would be fixed once we set the slope and intercept, which isn't as flexible as random forest, and also doesn't have the chance for different votings on different cases.

Question 23: Randomly pick a tree in your random forest model (with maximum depth of 4) and plot its structure. Which feature is selected for branching at the root node? What can you infer about the importance of features? Do the important features match what you got using p-values from linear regression?

Bike Dataset

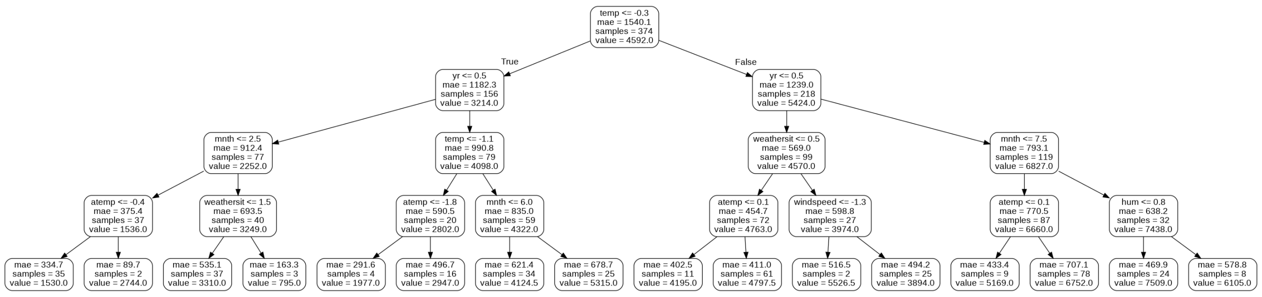


Figure 46: Tree sample from the bike dataset

The feature "temp" is selected to the the root node. It can be inferred that it has the highest importance of all. This matches what we got using p-values from linear regression.
We can also pull the feature importance data from the model to verify:

Feature	temp	year	atemp	season	hum	mnth	windspeed	weathersit
Importance	0.27	0.26	0.18	0.1	0.07	0.05	0.04	0.02

Table 23: Bike Dataset Feature Importance

Suicide Dataset

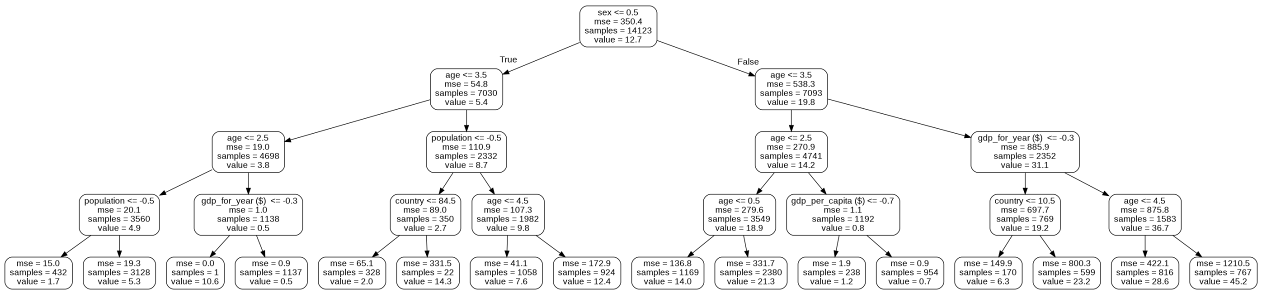


Figure 47: Tree sample from the suicide dataset

The feature "sex" is selected as the root node, which infers that it has a high level of relevance to our prediction. This satisfies our result from linear regression. Recall from our previous data inspection, where we saw that male's suicide rate is higher than female's. Our root node is determining whether sex is female or male, by screening the important features first, random forest model is efficient.
Pull the feature importance data from the model to verify:

Feature	country	age	population	sex	gdp_for_year	gdp_per_capita	year	generation
Importance	0.21	0.2	0.17	0.15	0.12	0.09	0.05	0.01

Table 24: Suicide Dataset Feature Importance

Video Dataset
We'll do a comparison with video dataset with and without one hot encoded feature, and see different trees.

In the first tree where we applied OHE data, x1_h264 is selected as the root node, which belongs to our feature "o_codec". And in the second tree, the root node is o_width. In our linear regression analyses, the most significant features with very small p-value are: o_codec_h264, o_framerate, o_bitrate, o_width. The result aligns with our analysis.

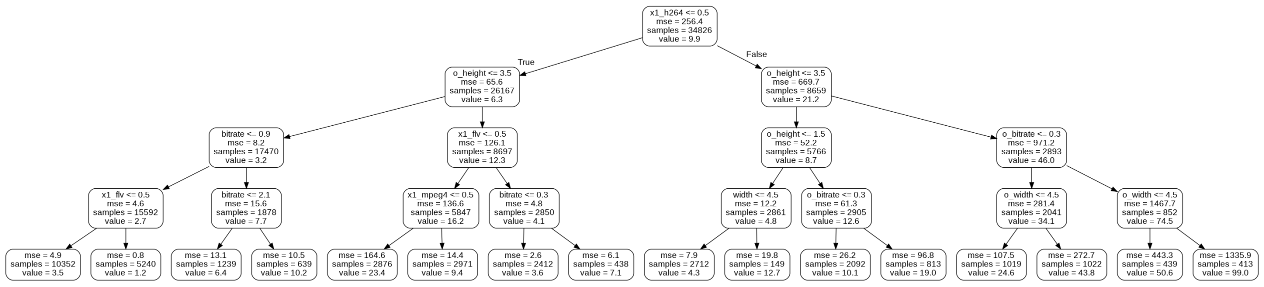


Figure 48: Tree sample from the video dataset (with one hot encoded feature)

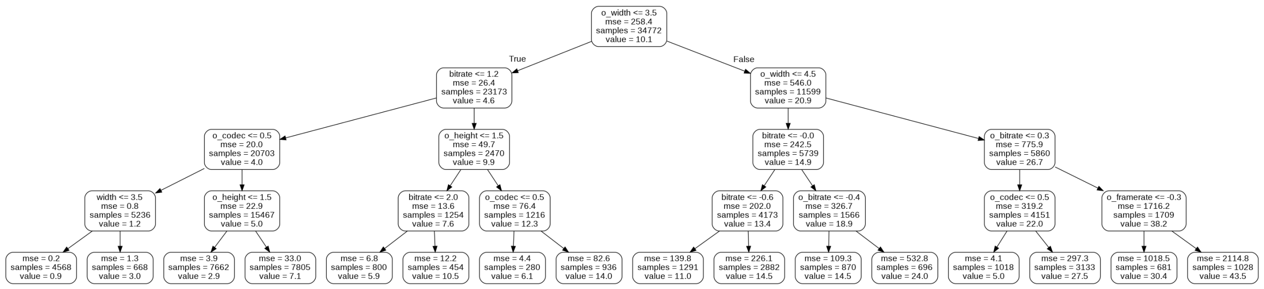


Figure 49: Tree sample from the video dataset (without one hot encoded feature)

Compare the most 5 important features in the two differently processed dataset, the feature name with its importance score is as follows:

with OHE	without OHE
`o_codec_h264` 0.25	`o_codec` 0.42
`o_width` 0.18	`o_width` 0.17
`o_bitrate` 0.15	`o_height` 0.15
`o_height` 0.15	`o_bitrate` 0.12
`o_codec_flv` 0.06	`o_framerate` 0.04

Table 25: Best 5 Features in Video dataset

LightGBM, CatBoost and Bayesian Optimization

For this part, we choose the video transcoding time dataset to experiment on.

Question 24: Experiment and determine the important hyperparameters for LightGBM and CatBoost, along with a proper search space for the tuning of these parameters.

We select the below hyperparameters with its search space to tune for the models:
LightGBM
num_leaves: control the complexity of tree model. As LightGBM uses leaf-wise tree, it can go much deeper than depth-wise tree and cause overfitting. For example, for a depth-wise tree of max depth 6, if we set the number of leaves to 2^6-1=63, the leaf-wise model may cause overfitting, as we can append a new node to any exisiting leaf node without the depth restriction; we can set number of leaves to 30 or 40 under this case. Thus, number of leaves can be tuned with maximum depth together. Default value 31.
Here we will elaborate on how we narrowed down the search space initially for later tuning. Take this feature for example, we tested different values: 10,50,100,500,800,1000; and found that after the value exceeds 500, the performance doesn't change. (Thus we defined the search scope for this feature from 2 to 500). The search space for the rest of the features were defined in a similar approach, but as we grew more sense about the earlier features, we used for loops to test on different combinations and observed how the model score would vary accordingly.
max_depth: limits the max depth of tree model. Default value is -1 which is no limit. (We tested from 5 to 50).
min_data_in_leaf: minimum number of observations that must fall into a tree node for it to be added. This feature as well as min_sum_hessian_in_leaf can be used to deal with overfitting. We'll pick one of them to include in testing. (We will sweep the value from 0 to 200).
subsample: specifies the percentage of rows used during each tree building iteration, randomly select rows in data to fit each tree. Default 1. (Our search space is 0.5 to 1).
subsample_freq: frequency for bagging. Default 0 means disable bagging. k means perform bagging at every k iteration. (We tested 0 to 10).
max_bin: default value 255, the max number of bins to bucket feature values into. (We tested from 10 to 1000).
lambda_l1/l2: L1/L2 regularization term on weights. (We tested 1e-7 to 10 on log-based for L1, and 1e-7 to 100 log-based for L2).
num_iterations: the number of trees to build. Model accuracy increases when there's more trees, but it'll take longer training time and may cause overfitting. Larger num_iterations could go with smaller learning_rate. (Our search scope is 10 to 1000)
learning_rate: default value 0.1, how quickly the error is corrected from each tree to the next. (Search space from 0.001 to 1 log-based)

CatBoost
Similarly, the search scope for CatBoost is defined.

Hyperparameters	Search Scope	Meaning
learning_rate	Real(0.001, 1.0, 'log-uniform')	how quickly error is corrected
n_estimators	Integer(10,1000)	number of boosted trees to fit
depth	Integer(4,10)	depth of tree
random_strength	Real(1e-7, 1.0, 'log-uniform')	randomness for scoring splits
bagging_temperature	(0.0, 1.0)	settings of the Bayesian bootstrap
border_count	(1,100)	splits for numerical features
l2_leaf_reg	Real(1e-7, 100, 'log-uniform')	L2 regularization

Table 26: Search Scope for CatBoost

Question 25: Apply Bayesian optimization to search for good hyperparameter combinations in your search space. Report the best hyperparameter found and the corresponding RMSE, for both algorithms.

In grid search and random search, we would try all the possible hyperparameter combinations within some ranges, which would not be efficient when we are tuning large search space hyperparameters. Each experiment is independent of each other, and the current information can not be used to improve the next.
Byesian optimization is smart, and can use the result from previous iteration to decide which hyperparameter value should be the next to test on. After applying Bayesian optimization on the search space we defined, the best hyperparameters and corresponding RMSE for both algorithms are:

	LightGBM	CatBoost

Best Hyperparameter	num_leaves=131
	max_depth=20
	min_data_in_leaf=108
	subsample=0.8538
	subsample_freq=4
	max_bin=813
	lambda_l1 (reg_alpha)=0.661
	lambda_l2 (reg_lambda)=6.032
	num_iterations=468
	learning_rate=0.056
		bagging_temperature=0.7206
		border_count=76
		depth=7
		l2_leaf_reg=7.969
		learning_rate=0.072
		n_estimators=715
		random_strength=0.00314

Training RMSE	1.124	1.455
Testing RMSE	3.729	3.971

Table 27: Model Results

Question 26: Interpret the effect of the hyperparameters using the Bayesian optimization results.

LightGBM:
Helps with performance: num_iterations, num_leaves, max_depth, learning_rate
Helps with regularization: lambda_l1, lambda_l2, subsample, max_bin, min_data_in_leaf
Affect fitting efficiency: num_leaves, max_depth
We'll select three hyperparameters among them to analyze. Applying the result from the Bayesian optimization, keep all other hyperparameter values unchanged, sweep the selected hyperparameter and observe its performance, which is indicated by training rmse and testing rmse.

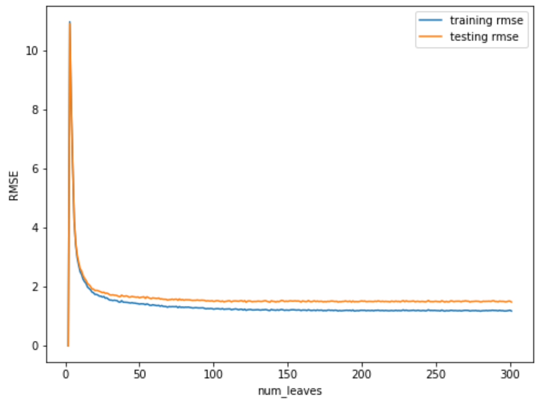


Figure 50: rmse vs. num_leaves

From the plot of rmse vs. num_leaves, we can see that as num_leaves becomes large, the testing rmse tends to deviate from training rmse more. Training rmse gets better, but testing rmse isn't as well due to overfitting. The overall performance gets better when the number increases, as the rmse is decreasing.

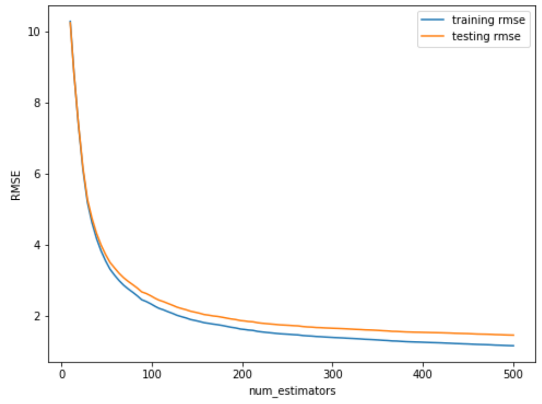


Figure 51: rmse vs. num_estimators

The hyperparameter num_estimators also helps with overall performance.

CatBoost:

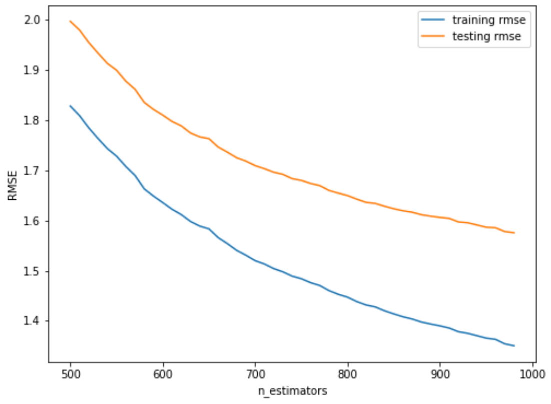


Figure 52: rmse vs. num_estimators

From the figure, both training and testing rmse decreases as n_estimator increases, and the gap between training and testing rmse also increases. This suggests that this features help with performance, but can cause overfitting when its value is large.

Evaluation

Question 27: Perform 10-fold cross-validation and measure average RMSE errors for training and validation sets. Why is the training RMSE different from that of validation set?

	Bike		Suicide		Video	
RMSE	Training	Testing	Training	Testing	Training	Testing
Linear Regression	990.567	989.841	14.858	14.852	11.856	11.851
Polynomial Regression	577.186	993.383	7.643	14.107	2.557	7.523
Neural Network	420.914	831.823	6.993	7.852	3.122	4.932
Random Forest	264.356	856.699	13.089	14.310	0.652	4.364
LightGBM	N/A		N/A		1.124	3.729
CatBoost	N/A		N/A		1.455	3.971

Table 28: Different Models Training RMSE vs. Testing RMSE

It can be seen from the table that training RMSE is lower than testing RMSE, this is because of overfitting. When the model is flexible, the training process adapts the model to the training data too much, and thus lose the accuracy on the new testing data. Even though we've tuned the hyperparameters, the effect of overfitting was not eliminated.

Question 28: For random forest model, measure "Out-of-Bag Error" (OOB) as well. Explain what OOB error and R^2 score means.

Dataset	bike	suicide	video
OOB Error	0.10765	0.48747	0.00883

Table 29: Out-of-Bag Error on Random Forest Model

Out-of-Bag Error:
We randomly selected samples from the original dataset with the same size, creating a bootstrapped dataset. The out-of-bag dataset are the entries that was left out from the bootstrapped dataset. Since the out-of-bag data wasn't used to create the trees, it might not be correctly predicted. The proportion of out-of-bag samples that were incorrectly labeled is the "OOB Error".
R^2 score: R^2 measures how close the data is fitted to the regression model. It quantifies the difference by calculting [val(mean)-val(model)]/val(mean)