CIS 5450 – Final Project Deliverable – Blog Post (Medium)

Team Members – Priyanka Agarwal, Purvansh Jain, Yash Agrawal

Link to Blog:

https://medium.com/@priyanka_60002/cis-5450-final-project-blog-22f1483fd424

P

Priyanka Agarwal

Dec 14 · 6 min read · Listen

# CIS 5450 — Final Project Blog

Team Members: Priyanka Agarwal, Purvansh Jain, Yash Agrawal

## Price Prediction of Internet Service Offers and Plans

Data Set Used — Internet Service Offers Dataset

Link — https://www.kaggle.com/datasets/michaelbryantds/internet-speeds-and-prices

. . .

## ABSTRACT

In this project, we want to perform a comprehensive predictive analysis of how internet service prices fluctuate across different providers over a particular demographic, bandwidth, geographic location, and other features.

It is a common notion that ISPs don't alter their plans for downtrodden sections of society as it is difficult to segregate backward sections. Analysis of this data will be useful to highlight the key features of an ISP plan and how different strategies can suit other customers.


Through the results of our analysis, we want to compile and compare the offers provided by different service providers according to varied characteristics. This can thus aid the individuals in the customer pool to make better and informed choices.

Since this is a regression-based task (Prediction of Internet service plan prices), we plan to use Supervised Regression techniques like Linear Regression, SVMs, Decision Trees, Random forests, GBM, XG-Boost, etc.

. . .

## ABOUT THE DATASET

In this first part, we mounted the dataset we obtained from Kaggle (link shared on top of the blog) on the drive and then explored its nature — the number of tables, the different ISPs, and the columns in each table.

The dataset consists of five CSV files and each file contains a different internet service provider except for AT&T which has two files. The other files are for Earthlink, Centurylink, and Verizon. The following table contains descriptions of the features in the dataset.

| column | description |
|---|---|
| address_full | The complete postal address of a household we searched. |
| incorporated_place | The incorporated city that the address belongs to. |
| major_city | The city that the address is in. |
| state | The state that the address is in. |
| lat | The address's latitude. From OpenAddresses or NYC Open Data. |
| lon | The address's longitude. From OpenAddresses or NYC Open Data. |
| block_group | The Census block group of the address, as of 2019. From the Census Geocoder API based on lat and lon. |
| collection_datetime | The Unix timestamp that the address was used to query the provider's website. |
| provider | The internet service provider. |
| speed_down | Cheapest advertised download speed for the address. |
| speed_up | Cheapest advertised upload speed for the address. |
| speed_unit | The unit of speed. This is always in megabits per second (Mbps). |

| | |
|---|---|
| `price` | The cost in USD of the cheapest advertised internet plan for the address. |
| `technology` | The kind of technology (fiber or non-fiber) used to serve the cheapest internet plan. |
| `package` | The name of the cheapest internet plan. |
| `fastest_speed_down` | The advertised download speed of the fastest package. This is usually the same as the cheapest plan if the `speed_down` is less than 200 Mbps. |
| `fastest_speed_price` | The advertised upload speed of the fastest internet package for the address. |
| `fn` | The name of the file of API responses where this record was parsed from. To be used for trouble shooting. API responses are hosted externally in AWS s3. |
| `redlining_grade` | The redlining grade, merged from Mapping Inequality based on the `lat` and `lon` of the adddress. |
| `race_perc_non_white` | The percentage of people of color (not non-Hispanic White) in the addresse's Census block group expressed as a proportion. Sourced from the 2019 5-year American Community Survey. |
| `median_household_income` | The median household income in the addresses' Census block group. Sourced from the 2019 5-year American Community |

| | |
|---|---|
| `income_lmi` | `median_household_income` divided by the city median household income (sourced from U.S. Census Bureau). |
| `income_dollars_below_median` | City median household income minus the `median_household_income`. |
| `ppl_per_sq_mile` | People per square mile is used to determine population density. Sourced from 2019 TIGER shape files from the U.S. Census Bureau. |
| `n_providers` | The number of other wired competitors in the addresses' Census block group. Sourced from FCC Form 477. |
| `internet_perc_broadband` | The percentage of the population that is already subscriped to broadband in an addresses' Census block group expressed as a proportion. |

To understand the columns in the dataset tables, we used some predefined methods like .dtypes, .describe, .columns, etc and in general, reviewed the metadata available on Kaggle. Some reference screenshots are as follows:

```
[ ]  #understanding column names
     att_df.columns
```

```
Index(['address_full', 'incorporated_place', 'major_city', 'state', 'lat',
       'lon', 'block_group', 'collection_datetime', 'provider', 'speed_down',
       'speed_up', 'speed_unit', 'price', 'technology', 'package',
       'fastest_speed_down', 'fastest_speed_price', 'fn', 'redlining_grade',
       'race_perc_non_white', 'income_lmi', 'ppl_per_sq_mile', 'n_providers',
       'income_dollars_below_median', 'internet_perc_broadband',
       'median_household_income'],
      dtype='object')
```

## EXPLORATORY DATA ANALYSIS (EDA)

Our aim is to make sure every individual dataset has the same schema after preprocessing and cleaning steps so we can merge and do further analysis and manipulation of the complete data.

**Understanding packages and their respective fast/slow speends and cheap/coslty price**

```
#creating a temporary dataframe view with relevant columns
att_df[['package','fastest_speed_price','price','speed_down','fastest_speed_down']]
```

| | package | fastest_speed_price | price | speed_down | fastest_speed_down |
|---|---|---|---|---|---|
| **0** | Internet Basic 768kbps | 55.0 | 55.0 | 0.768 | 0.768 |
| **1** | Internet Basic 5 | 55.0 | 55.0 | 5.000 | 5.000 |
| **2** | Internet Basic 768kbps | 55.0 | 55.0 | 0.768 | 0.768 |
| **3** | Internet Basic 5 | 55.0 | 55.0 | 5.000 | 5.000 |
| **4** | AT&T FIBER—INTERNET 300 | 180.0 | 55.0 | 300.000 | 5000.000 |
| **...** | ... | ... | ... | ... | ... |
| **432298** | AT&T FIBER—INTERNET 300 | 80.0 | 55.0 | 300.000 | 1000.000 |
| **432299** | AT&T FIBER—INTERNET 300 | 80.0 | 55.0 | 300.000 | 1000.000 |
| **432300** | AT&T FIBER—INTERNET 300 | 80.0 | 55.0 | 300.000 | 1000.000 |
| **432301** | Internet 50 | 55.0 | 55.0 | 50.000 | 50.000 |
| **432302** | AT&T FIBER—INTERNET 300 | 80.0 | 55.0 | 300.000 | 1000.000 |

432303 rows × 5 columns

Understanding more about different packages through projections of relevant columns in data frames.

We compared columns in different datasets to understand what extra data was available among different ISP data tables.

```
[ ]  #comparing the columns of att_df and att_other_cities_df
     set(att_df.columns) - set(att_other_cities_df.columns)
```

```
{'income_dollars_below_median', 'income_lmi', 'n_providers', 'ppl_per_sq_mile'}
```

```
[ ]  set(att_other_cities_df.columns)  - set(att_df.columns)
```

```
{'availability_status', 'geoid'}
```

. . .

## CLEANING AND PREPROCESSING OF DATA

We cleaned and processed individual data frames so that we can merge them into a single data table. To do so, we performed the following steps:

- Checking column names.
- Handling redundancy due to space anomalies in packages.
- Checking for null values and imputing data as per appropriate fit.
- Dropping columns and rows irrelevant for our modeling stage.
- Checking data types and converting them to appropriate data types.

Handling ppl_per_sq_mile null values

+ Code + Text

```
[ ]  #finding the cityy where ppl_per_sq_mile is null
     att_df[att_df['ppl_per_sq_mile'].isnull()]['major_city'].value_counts()
```

```
oklahoma city    6463
Name: major_city, dtype: int64
```

We see that all the null rows have only oklahoma city as their major city. Hence we can do simple substitution by mean based on major city = Oklahoma city

```
[ ]  oklahoma_mean = att_df[att_df['major_city'] == 'oklahoma city']['ppl_per_sq_mile'].mean()
     att_df['ppl_per_sq_mile'].loc[att_df[att_df['ppl_per_sq_mile'].isnull()].index] = oklahoma_mean
```

**Columns to drop**

- speed_unit - since we know that the unit is always mbps
- redlining grade - since large number of rows are null
- address - since we do not require specific building addresses

```
[ ]  # dropping the above specified columns
     att_df.drop(columns = ['speed_unit','redlining_grade', 'address_full'],axis=1,inplace=True)
```

Checking if there are anomalies due spaces in package name like there was in att_df data

```
[ ]  att_other_cities_df.package.value_counts()
```

```
AT&T FIBER—INTERNET 300    8990
Internet 50                4996
Internet Basic 5           2207
Internet 25                2196
Internet 100               1858
Internet 18                1693
Internet 10                1572
Internet 75                1522
Internet Basic 768kbps     1003
Internet Basic 1.5          714
Name: package, dtype: int64
```

Checking for anomalies in median_household_income and dropping the rows

```
[ ]  att_other_cities_df[att_other_cities_df['median_household_income']<0]['median_household_income'].value_counts()
     att_other_cities_df.drop(att_other_cities_df[att_other_cities_df['median_household_income']<0].index,axis=0,inplace=True)
```

We cleaned and processed all the different tables for the different service providers— Verizon, Earthlink, CenturyLink, and AT&T. Post cleaning, we finally merged the cleaned tables into a single data frame.

```
final_merged_df = pd.concat([att_cleaned_df,earthlink_data,centurylink_data,verizon_data]).reset_index(drop=True)
final_merged_df
```

| | major_city | state | lat | lon | block_group | provider | speed_down | speed_up | price | technology | package | fastest_speed_down | fastest_speed_price | race_perc_non_white | income_inc | ppl_per_sq_mile | n_providers | income_dollars_below_median | internet_perc_broadband | median_household_income |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | huntsville | AL | 34.745179 | -86.696896 | 100800070223 | AT&T | 0.768 | 0.384 | 55.0 | Not Fiber | Internet Basic 768kbps | 0.768 | 55.0 | 0.475059 | 0.381744 | 512.090969 | 4.0 | 35091.0 | 0.527881 | 21667 |
| 1 | huntsville | AL | 34.747563 | -86.807140 | 100800070222 | AT&T | 5.000 | 1.000 | 55.0 | Not Fiber | Internet Basic 5 | 5.000 | 55.0 | 0.475059 | 0.381744 | 512.090969 | 4.0 | 35091.0 | 0.527881 | 21667 |
| 2 | huntsville | AL | 34.746504 | -86.805084 | 100800070222 | AT&T | 0.768 | 0.384 | 55.0 | Not Fiber | Internet Basic 768kbps | 0.768 | 55.0 | 0.475059 | 0.381744 | 512.090969 | 4.0 | 35091.0 | 0.527881 | 21667 |
| 3 | huntsville | AL | 34.747750 | -86.807075 | 100800070222 | AT&T | 5.000 | 1.000 | 55.0 | Not Fiber | Internet Basic 5 | 5.000 | 55.0 | 0.475059 | 0.381744 | 512.090969 | 4.0 | 35091.0 | 0.527881 | 21667 |
| 4 | huntsville | AL | 34.748505 | -86.695223 | 100800070222 | AT&T | 300.000 | 300.000 | 55.0 | Fiber | AT&T FIBER—INTERNET 300 | 5000.000 | 150.0 | 0.475059 | 0.381744 | 512.090969 | 4.0 | 35091.0 | 0.527881 | 21667 |
| ... | | | | | | | | | | | | | | | | | | | | |
| 1156983 | virginia beach | VA | 36.724616 | -75.937796 | 518100454121 | Verizon | 10.000 | 1.000 | 40.0 | Not fiber | HSI Up to 7.1 - 15 Mbps | 10.000 | 40.0 | 0.078313 | 1.273361 | 1695.0 | 2.0 | -21750.0 | 0.957447 | 99866 |
| 1156984 | virginia beach | VA | 36.712828 | -75.931552 | 518100454121 | Verizon | 10.000 | 1.000 | 40.0 | Not fiber | HSI Up to 7.1 - 15 Mbps | 10.000 | 40.0 | 0.078313 | 1.273361 | 1695.0 | 2.0 | -21750.0 | 0.957447 | 99866 |
| 1156985 | virginia beach | VA | 36.709837 | -75.930980 | 518100454121 | Verizon | 10.000 | 1.000 | 40.0 | Not fiber | HSI Up to 7.1 - 15 Mbps | 10.000 | 40.0 | 0.078313 | 1.273361 | 1695.0 | 2.0 | -21750.0 | 0.957447 | 99866 |
| 1156986 | virginia beach | VA | 36.754802 | -75.948176 | 518100454121 | Verizon | 10.000 | 1.000 | 40.0 | Not fiber | HSI Up to 7.1 - 15 Mbps | 10.000 | 40.0 | 0.078313 | 1.273361 | 1695.0 | 2.0 | -21750.0 | 0.957447 | 99866 |
| 1156987 | virginia beach | VA | 36.708564 | -75.930542 | 518100454121 | Verizon | 10.000 | 1.000 | 40.0 | Not fiber | HSI Up to 7.1 - 15 Mbps | 10.000 | 40.0 | 0.078313 | 1.273361 | 1695.0 | 2.0 | -21750.0 | 0.957447 | 99866 |

1156988 rows × 20 columns

final_merged_df

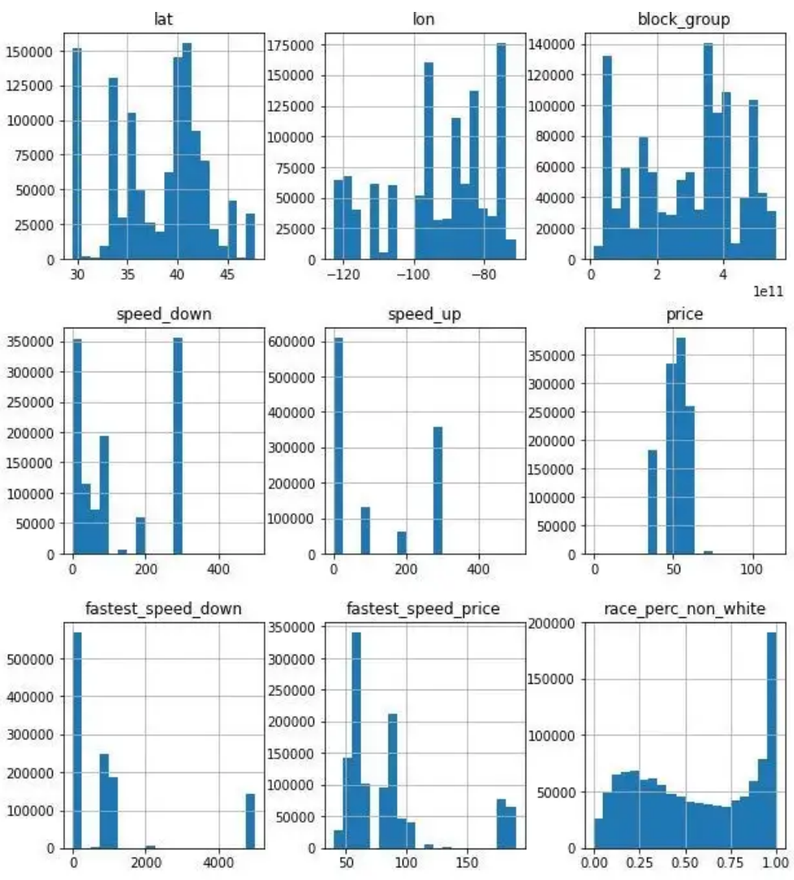
. . .

## DATA VISUALIZATION

We created some interesting visualizations to understand trends observed in our data — both numerical and categorical.

```
[ ]  categorical_columns = final_merged_df.loc[:,final_merged_df.dtypes==np.object].columns
     numerical_columns = list(set(final_merged_df.columns) - set(categorical_columns))
```

```
numerical_columns_df = pd.DataFrame(numerical_columns,columns = ['columns'])
fig, axis = plt.subplots(5,3,figsize=(10, 20))
final_merged_df.hist(ax=axis,bins = 20)
```

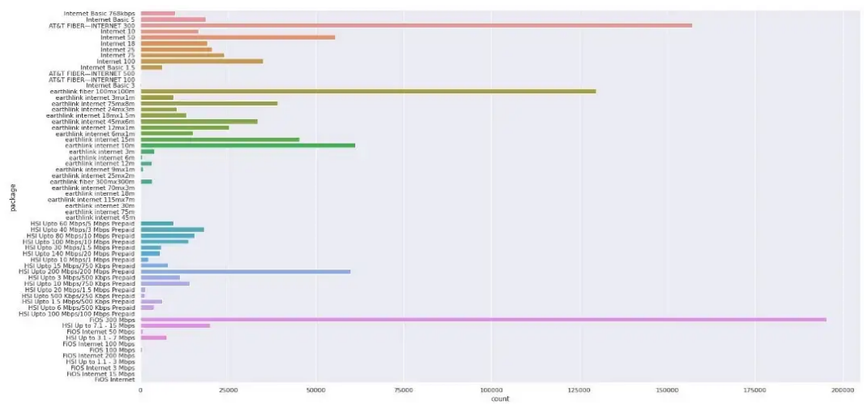



Key takeaways from visualizing numerical data:

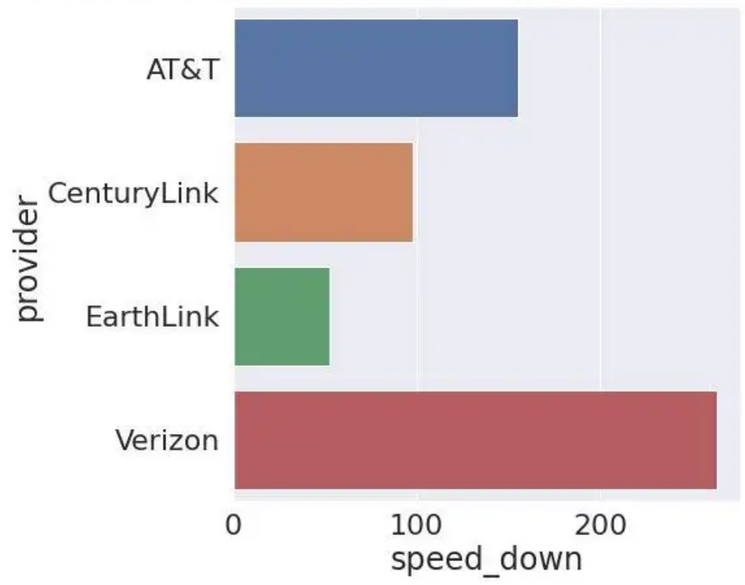
- We can see two different trends in downloading speed of the cheapest plan (down_speed). Either people go for high speeds like 300mbps or cheap plans like 50mbps.
- Exploring fastest download speed — Majorly people go for speeds around 1000mbps, and some people go for 4000mbps.
- Exploring price — our dependent variable demonstrates a normal distribution curve with a mean around 55usd.
- Income features show that there is a very small section of people who earn outrageously high. These are potential outliers.

```
[ ]  categorical_columns_df = pd.DataFrame(categorical_columns,columns = ['columns'])
     categorical_df = final_merged_df[list(categorical_columns_df['columns'].values)]
     sns.set(font_scale = 1)

     c = 1
     plt.figure(figsize = (20,50))
     for i in categorical_df.columns:
       plt.subplot(5, 1, c)
       plt.tight_layout()
       sns.countplot(data= categorical_df, y=i)

       c = c+1
```

Visualizing Categorical Data

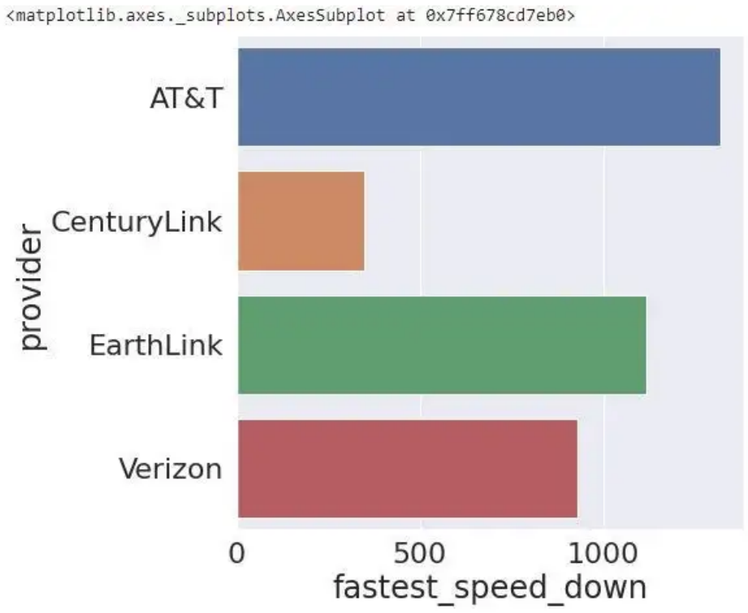



```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff678d9bfa0>
```



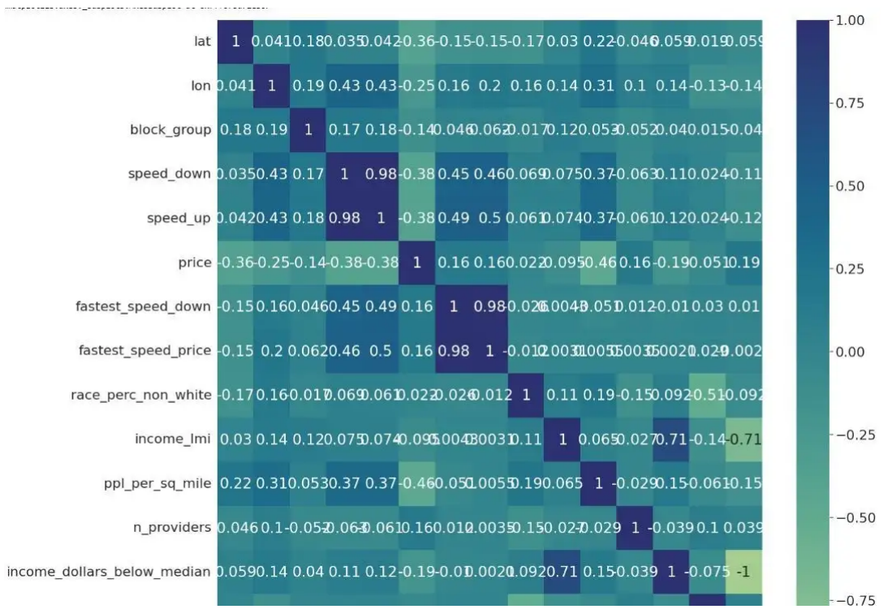

As seen Verizon has the average highest download speed and Earthlink has the lowest average download speed when it comes to the cheapest plans.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff678cd7eb0>
```




Surprisingly, when it comes to expensive plans, Centurylink has the lowest download speed and AT&T has the highest average download speed. We can see EarthLink dominates the category of the expensive plan compared to other providers despite EarthLink wasn't good when it came to cheaper plans.

We also plotted a heatmap and found out that none of the features is strongly correlated to our dependent variable price. Hence, there is no problem with multicollinearity.




heatmap


We included map visualization for each lat long in the USA with different providers. This helps us identify trends of prices for different localities in the entire country,

. . .

## MODELLING AND PREDICTION

We first started with Linear regression with and without regularization and followed it with more complex models like Tree models and Ensemble Models. We used R2, MAE, RMSE as our metrics for determining which model is best for prediction.

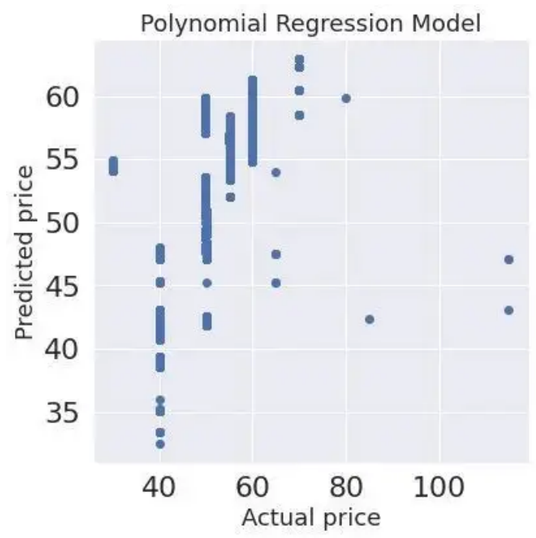
The below table demonstrates the results when we tried linear models (with/without regularization) and tried hyperparameter optimization using Grid Search, and Randomized Search to tackle the problem of bias-variance trade-off if any.

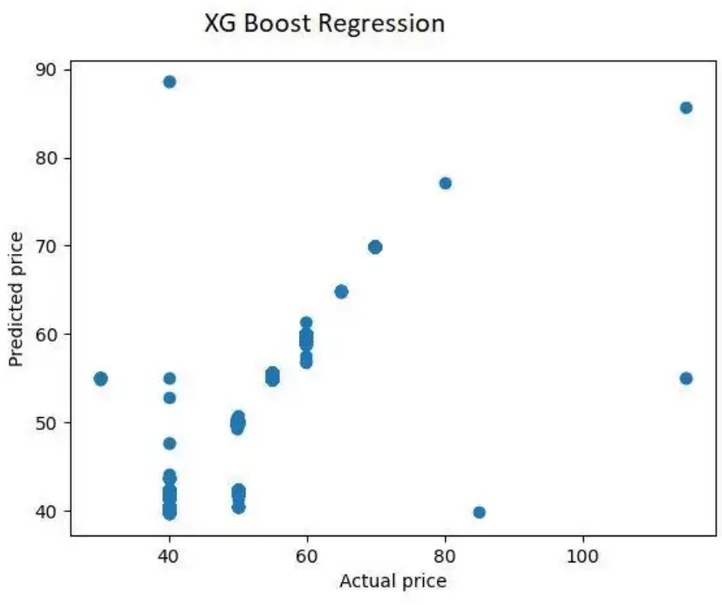
| | Model Name | R2 score | MAE | RMSE |
|---|---|---|---|---|
| 0 | Baseline Linear Regression | 0.785477946503195 | 2.414084156349595 | 2.996457816749097 |
| 0 | Polynomial Regression | 0.8667133741693545 | 1.59434274038019 | 2.3619216245648427 |
| 0 | Baseline Ridge Regression on scaled features | 0.7854779461764777 | 2.4140872314299027 | 2.9964578190309012 |
| 0 | Ridge Regression Grid search on scaled features, 5 folds | 0.7854779461764777 | 2.4140872314299027 | 2.9964578190309012 |
| 0 | Ridge Regression Grid search on scaled features 20 folds | 0.7854779446496987 | 2.414099531521399 | 2.9964578296939752 |
| 0 | Baseline Lasso Regression on scaled features | 0.5326991406015515 | 3.6065146432372046 | 4.422526921397267 |
| 0 | Lasso Regression Grid search on scaled features 20 folds | 0.7854779211932397 | 2.414093169388218 | 2.996457993514633 |
| 0 | Baseline Elasticnet Regression on scaled features | 0.5147539224408982 | 3.553359597250637 | 4.506643580906438 |
| 0 | Elasticnet Regression Grid search on scaled features 10 folds | 0.7854777099272492 | 2.4141918676693206 | 2.996459469002642 |
| 0 | Elasticnet Regression Grid search on scaled features 20 folds | 0.7837431255490017 | 2.4179530938470646 | 3.006192449853415 |

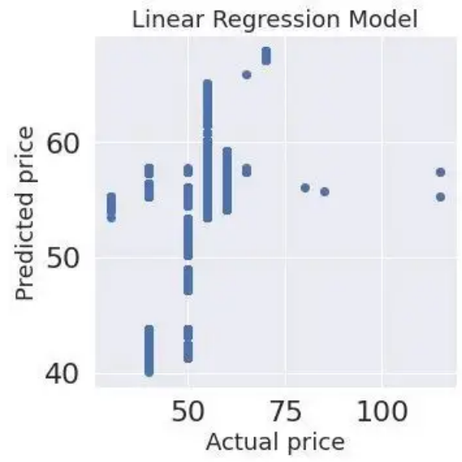
Thus we can conclude that all linear, ridge, lasso, and elastic net can only deduce patterns up to a certain score. The polynomial model was still better than all these models (R2 ~ 86/87%).

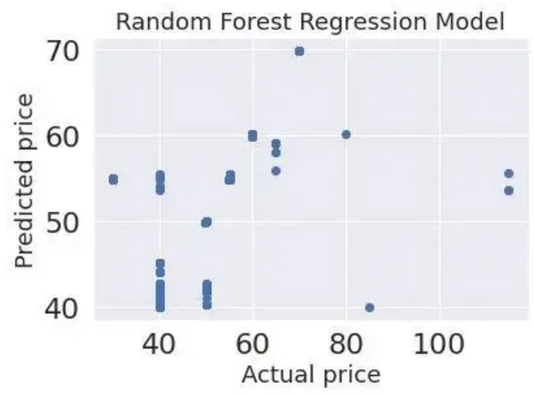
We tried further complex models, particularly ensemble methods — Bagging (Random Forest) and Boosting models(Adaboost, GBM, XGBoost). We also trained neural networks to see how they performed on tabular data. The performance of these algorithms is given in the table below under “Final Result Metrics”

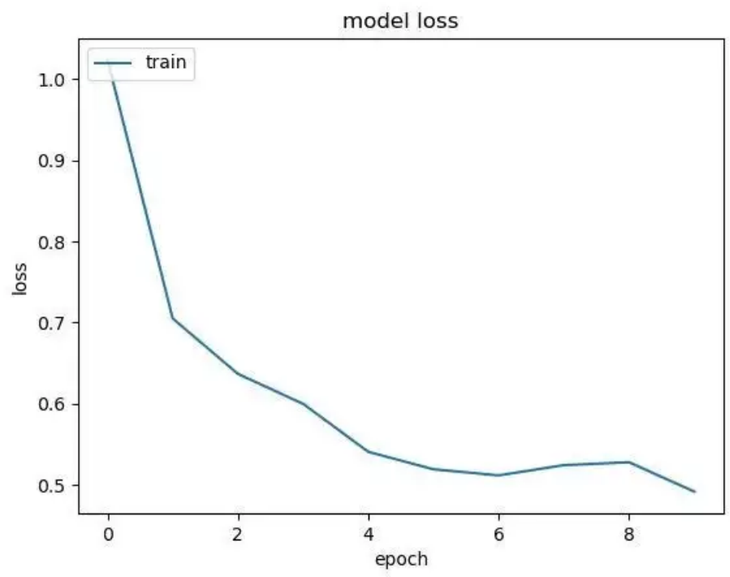
The below figures show the scatter plot of actual and predicted labels for different models. Just visual inspection of these helps us understand how well tree-based models perform better than linear models. For any given x, the predictions are less scattered away for tree models and more scattered away for linear models.






**FINAL RESULT METRICS:**

| | Model Name | R2 score | MAE | RMSE |
|---|---|---|---|---|
| 0 | Baseline Linear Regression | 0.785477946503195 | 2.414084156349595 | 2.996457816749097 |
| 1 | Polynomial Regression | 0.8667133741693545 | 1.59434274038019 | 2.3619216245648427 |
| 2 | Baseline Ridge Regression on scaled features | 0.7854779461764777 | 2.4140872314299027 | 2.9964578190309012 |
| 3 | Ridge Regression Grid search on scaled features, 5 folds | 0.7854779461764777 | 2.4140872314299027 | 2.9964578190309012 |
| 4 | Ridge Regression Grid search on scaled features 20 folds | 0.7854779446496987 | 2.414099531521399 | 2.9964578296939752 |
| 5 | Baseline Lasso Regression on scaled features | 0.5326991406015515 | 3.6065146432372046 | 4.422526921397267 |
| 6 | Lasso Regression Grid search on scaled features 20 folds | 0.7854779211932397 | 2.414093169388218 | 2.996457993514633 |
| 7 | Baseline Elasticnet Regression on scaled features | 0.5147539224408982 | 3.553359597250637 | 4.506643580906438 |
| 8 | Elasticnet Regression Grid search on scaled features 10 folds | 0.7854777099272492 | 2.4141918676693206 | 2.996459469002642 |
| 9 | Elasticnet Regression Grid search on scaled features 20 folds | 0.7837431255490017 | 2.4179530938470646 | 3.006192449853415 |
| 10 | Baseline Decision Tree Regression | 0.93143887909494 | 0.569121412008691 | 1.6940405064199606 |
| 11 | Decision Tree Regression Randomized 1 | 0.9295070042417883 | 0.5776583181179004 | 1.7176915494905312 |
| 12 | Decision Tree Regression Grid1 | 0.9283813838970363 | 0.583581438459767 | 1.7299947341413182 |
| 13 | Baseline Random Forest Regression | 0.9314170828129452 | 0.5691960452432564 | 1.6942604476987246 |
| 14 | Random Forest Regression Randomized1 | 0.9305432495681365 | 0.5711553863432476 | 1.7050197964267002 |
| 15 | Baseline Ada Boost Regression | 0.8382134753539906 | 1.6844882703931612 | 2.6022161532008568 |
| 16 | Baseline GBM Regression | 0.9283637907370174 | 0.7151859850755389 | 1.7315637830631434 |
| 17 | Baseline XGBoost Regression | 0.9285428470735395 | 0.7128864444332323 | 1.7293983875471801 |
| 18 | XGBoost Regression Randomized 2 fold | 0.9287151719754384 | 0.5915719169978657 | 1.7259585907167008 |
| 19 | Artificial Neural network | 0.9110563665677295 | 0.40745178767633744 | 1.9294326740805283 |
| 20 | Random Forest Regression Randomized1 | 0.9305432495681365 | 0.5711553863432476 | 1.7050197964267002 |

R2 Score, MAE, RMSE for each model

The above table shows the metric scores for R2, MAE, and RMSE for all the model versions that we have trained. Tree-based models have outperformed all other models as they explain most of the variance in the model. Neural networks tend to perform better than linear models for tabular data, but it's not considered the best model family. That is what we have observed from our results. Neural nets have achieved better accuracy than linear models but tree-based models were a tad bit better.

## CONCLUSION

So far in our project, we have utilized capabilities of different models like Linear models with/without regularization, tree-based models, ensemble methods, Neural networks, etc. to carry out comprehensive price regression. It is a common notion that ISPs don't alter their plans for downtrodden sections of society as it is difficult to segregate sections of society according to economic conditions. Therefore, the analysis and inferences from modeling on this data can be helpful to highlight trends in prices for different features like download speed, geographical locations, etc, for internet offer plans by different Internet Service Providers. Potential customers can use the results of this extensive analysis to aid their decision-making process when choosing a plan according to the location of their residence.