Machine Learning Project Phase 2

Predicting Revenue-Related Metrics using Google ads data

Name: Minh Phan

Student number: s3335814

Predicting Revenue-Related Metrics using Google ads data

June 9, 2019

1 Predicting revenue-related metric

The objective of this case study is to fit and compare a few different reggessors to predict the revenue-related metric-continuous variable. The descriptive features include 15 numeric and 6 nominal categorical features, two of which were self generated. The full dataset contains about 64K observations (after ignoring all the na values). Some of the numeric variables has been transformed using log transformation.

This report is organized as follows:

Section 2 (Overview) outlines our methodology.

Section 3 (Data Preparation) summarises the data preparation process and our model evaluation strategy.

Section 4 (Hyper-parameter Tuning) performs the hyper-parameter tuning process for each classification algorithm.

Section 5 (Performance Comparison) presents model performance comparison results.

Section 6 (Limitations) discusses a limitations of our approach and possible solutions.

Section 7 (Summary) provides a brief summary of our work in this project.

2 Overview

2.0.1 Methodology

We build the following regressors to predict the target feature:

- K-Nearest Neighbours (KNN),
- Decision trees (DT), and
- Naive Bayes (NB) and some other linear models.

Our modeling strategy begins by transforming the full dataset cleaned from Phase I. This transformation includes encoding categorical descriptive features as numerical and then scaling of the descriptive features. We first randomly sample 5K rows from the full dataset and then split this sample into training and test sets with a 70:30 ratio. This way, our training data has 3500 rows and test data has 1500 rows.

Before fitting a particular model, we selected only the features with high correlation with the target variable.

The categorical features are then transformed using dummy-variable. Subsequently, these features are ranked by using F-score; we have considered 14 features and the full set of features.

For both KNN and DT regressors, we tune the parameters by running and comparing the Rooted Mean Squared Error(RMSR) and choose the best parameters. With linear models, we have fitting different linear models to compare with Naive Bayes.

Once the model with the lowest RMSE was selected, we also did a visual inspection of the residuals distribution.

3 Data Preparation

3.1 Loading Dataset

We load the dataset from a saved file from Phase 1.

```
In [1]: import pandas as pd
        import numpy as np
        import warnings
        warnings.filterwarnings("ignore")
        import numpy as np
        import pandas as pd
```

```
In [2]: ad=pd.read_csv("ad_data.csv")
```

3.2 Dealing with missing values

```
In [3]: ad.isna().sum()
```

```
Out[3]: Unnamed: 0         0
        companyId          0
        countryId          0
        deviceType         0
        dow                0
        price1         82223
        price2         82135
        price3         82135
        ad_area            0
        ad_ratio           0
        requests       71639
        impression     71639
        cpc            71678
        ctr            71666
        viewability        0
        ratio1             0
        ratio2             0
        ratio3             0
        ratio4             0
        ratio5             0
        y                  0
        adSeen             0
        adType             0
```

```
log_y              0
dtype: int64
```

```
In [4]: ad=ad.dropna()
```

```
In [5]: print(ad.dtypes)
```

```
Unnamed: 0       int64
companyId        int64
countryId        int64
deviceType       int64
dow             object
price1         float64
price2         float64
price3         float64
ad_area        float64
ad_ratio       float64
requests       float64
impression     float64
cpc            float64
ctr            float64
viewability    float64
ratio1         float64
ratio2         float64
ratio3         float64
ratio4         float64
ratio5         float64
y              float64
adSeen          object
adType          object
log_y          float64
dtype: object
```

We also delete the 'Unnames: 0' variable since it represents the indices. In addition we also exclude y- the orginal target variable, because we will use the log-transformed of it as the target variable.

```
In [6]: ad= ad.drop("Unnamed: 0",axis=1)
        ad= ad.drop("log_y",axis=1)
```

```
In [7]: ad.corr()['y'].sort_values()
```

```
Out[7]: requests      -0.283419
        impression    -0.197041
        ratio5        -0.185175
        companyId     -0.128421
        ratio2        -0.114704
        countryId     -0.044929
```

```
ratio4         -0.039350
ad_area        -0.033005
ad_ratio        0.028364
cpc             0.058204
deviceType      0.119916
ratio1          0.149058
ratio3          0.152404
ctr             0.314249
viewability     0.321137
price1          0.329727
price2          0.407400
price3          0.407400
y               1.000000
Name: y, dtype: float64
```

As we discussed in Phase 1, we will select only features that are highly correlated with the y value. In this instance, we will ignore features with the absolute value of the correlation that are less than 0.1; this includes ad_ratio,countryId,ratio4 and ad_area.

```
In [8]: ad= ad.drop("ad_ratio",axis=1)
        ad= ad.drop("countryId",axis=1)
        ad= ad.drop("ratio4",axis=1)
        ad= ad.drop("ad_area",axis=1)
        ad= ad.drop("cpc",axis=1)
```

3.3 Encoding Categorical Descriptive Features

Since some of the descriptive features are nominal, we perform one-hot-encoding. Furthermore, since we plan on conducting feature selection, we define q dummy variables for a categorical descriptive variable with q levels.

```
In [9]: ad['dow'] = ad['dow'].astype(str)
        ad['adSeen']=ad['adSeen'].astype(str)
        ad['adType']=ad['adType'].astype(str)
        ad['deviceType']=ad['deviceType'].astype(str)
        ad['companyId']=ad['companyId'].astype(str)

In [10]: ad_hot= pd.get_dummies(ad, columns=['deviceType','dow','adSeen','adType','companyId']

In [11]: ad_hot.head(6)

Out[11]:      price1    price2    price3  requests  impression       ctr  viewability  \
         0  -2.207275 -0.510826 -0.510826  9.266437    8.377931 -6.812445       0.0557
         3  -0.248461  0.778682  0.778682  8.510571    6.945051 -6.265901       0.1883
         4   1.517323  1.517893  1.517893  7.304516    3.295837 -0.811030       0.8750
         22 -1.108663  0.378436  0.378436  9.495970    8.551401 -6.377127       0.4939
         28 -4.605170 -1.215371 -1.215371  8.150468    6.665684 -6.645391       0.2291
         29 -1.832581 -0.466171 -0.466171  7.878913    6.429719 -5.339139       0.5683
```

```
    ratio1  ratio2  ratio3       ...       adSeen_seen  adType_banner  \
0   0.8630  0.4811  0.0646       ...                 0              1
3   0.6474  0.9595  1.0000       ...                 0              1
4   1.0000  1.0000  1.0000       ...                 1              0
22  0.2586  0.9656  0.1405       ...                 0              1
28  0.7529  0.6102  0.0408       ...                 0              1
29  0.6435  0.9661  0.0339       ...                 1              1

    adType_half-page  adType_regtangular  companyId_126  companyId_157  \
0                  0                   0              0              0
3                  0                   0              0              0
4                  1                   0              0              0
22                 0                   0              0              0
28                 0                   0              0              0
29                 0                   0              0              0

    companyId_159  companyId_40  companyId_43  companyId_95
0               0             0             0             1
3               0             0             1             0
4               0             0             1             0
22              0             0             1             0
28              0             0             1             0
29              0             0             1             0

[6 rows x 34 columns]
```

3.4 Subsetting and Scaling of Features

Because of the restriction of time and computer power, we selected a subset of 5000 observations.In addition, we split the subset into 'Data' and 'Target' for the training process. Finally, we perform a min-max scaling of the descriptive features.

```
In [12]: ad_subset = ad_hot.sample(n=5000)
```

```
In [13]: ad_subset_copy=ad_subset.drop(columns = 'y')
         Data = ad_subset_copy.values

         target = ad_subset['y']
```

```
In [14]: ad_subset_copy.shape
```

```
Out[14]: (5000, 33)
```

```
In [15]: from sklearn import preprocessing
```

```
In [16]: Data = preprocessing.MinMaxScaler().fit_transform(Data)
```

3.5 Feature Selection & Ranking

We will have a look at all the features using F-score methods. We'll select the top 45 and then 90.

```
In [17]: from sklearn import feature_selection as fs
```

```
In [18]: fs_fit_fscore = fs.SelectKBest(fs.f_classif, k=14)
         fs_fit_fscore.fit_transform(Data, target)
         fs_indices_fscore = np.argsort(fs_fit_fscore.scores_)[::-1][0:14]
         fs_indices_fscore
```

```
Out[18]: array([32, 30, 28, 26, 22, 29, 23,  9, 31,  1,  2, 16, 12, 10])
```

```
In [19]: best_features_fscore = ad_subset_copy.columns[fs_indices_fscore].values
         best_features_fscore
```

```
Out[19]: array(['companyId_95', 'companyId_40', 'companyId_157',
                'adType_regtangular', 'adSeen_partly-seen', 'companyId_159',
                'adSeen_seen', 'ratio3', 'companyId_43', 'price2', 'price3',
                'dow_Saturday', 'deviceType_2', 'ratio5'], dtype=object)
```

```
In [20]: feature_importances_fscore = fs_fit_fscore.scores_[fs_indices_fscore]
         feature_importances_fscore
```

```
Out[20]: array([       inf,        inf,        inf, 3.48301086, 3.13302309,
                2.3574041 , 2.28776037, 2.28561996, 2.19061267, 2.14315083,
                2.14315083, 1.87045921, 1.82414464, 1.58763842])
```

```
In [21]: import altair as alt
         alt.renderers.enable('notebook')
```

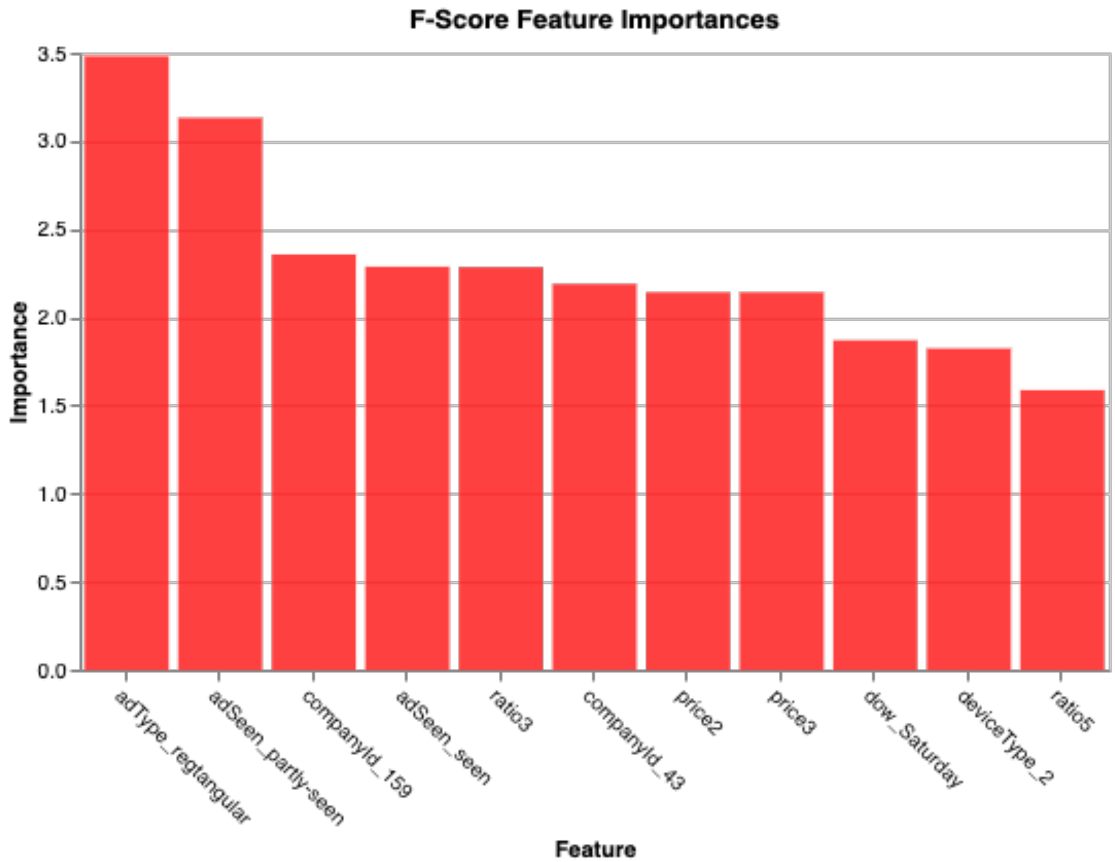
```
Out[21]: RendererRegistry.enable('notebook')
```

```
In [22]: def plot_imp(best_features, scores, method_name, color):

             df = pd.DataFrame({'features': best_features,
                                'importances': scores})

             chart = alt.Chart(df,
                               width=500,
                               title=method_name + ' Feature Importances'
                              ).mark_bar(opacity=0.75,
                                         color=color).encode(
                 alt.X('features', title='Feature', sort=None, axis=alt.AxisConfig(labelAngle=4
                 alt.Y('importances', title='Importance')
             )

             return chart
```

```
In [23]: plot_imp(best_features_fscore, feature_importances_fscore, 'F-Score', 'red')
```

```
<vega.vegalite.VegaLite at 0x7fd60a3e6f28>
```

Out[23]:



In [24]:
```
fs_fit_fscore = fs.SelectKBest(fs.f_classif, k=30)
fs_fit_fscore.fit_transform(Data, target)
fs_indices_fscore = np.argsort(fs_fit_fscore.scores_)[::-1][0:30]
fs_indices_fscore
```

Out[24]:
```
array([32, 30, 28, 26, 22, 29, 23,  9, 31,  1,  2, 16, 12, 10, 24,  6,  4,
       25,  0,  5,  3, 17, 11, 18, 13,  8, 14, 15, 20, 19])
```

In [25]:
```
best_features_fscore = ad_subset_copy.columns[fs_indices_fscore].values
best_features_fscore
```

Out[25]:
```
array(['companyId_95', 'companyId_40', 'companyId_157',
       'adType_regtangular', 'adSeen_partly-seen', 'companyId_159',
       'adSeen_seen', 'ratio3', 'companyId_43', 'price2', 'price3',
       'dow_Saturday', 'deviceType_2', 'ratio5', 'adType_banner',
       'viewability', 'impression', 'adType_half-page', 'price1', 'ctr',
```

```
'requests', 'dow_Sunday', 'deviceType_1', 'dow_Thursday',
'deviceType_3', 'ratio2', 'dow_Friday', 'dow_Monday',
'dow_Wednesday', 'dow_Tuesday'], dtype=object)
```

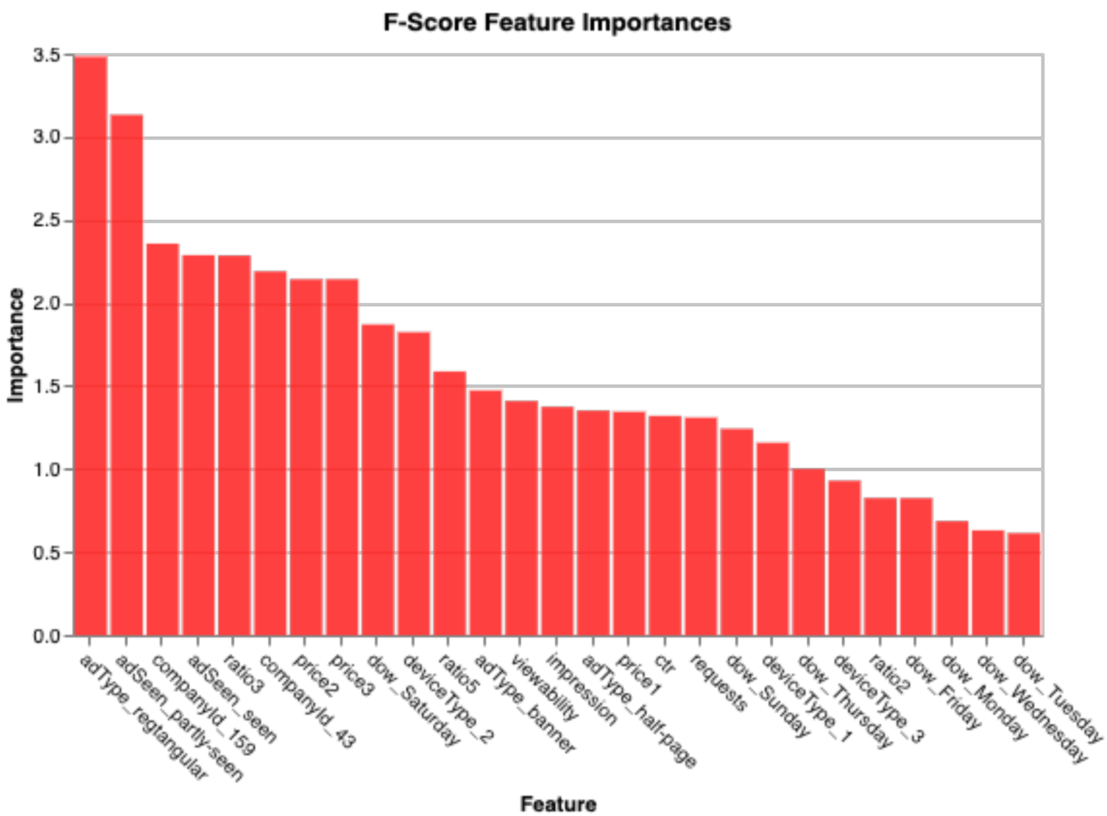
In [26]:
```
feature_importances_fscore = fs_fit_fscore.scores_[fs_indices_fscore]
feature_importances_fscore
```

Out[26]:
```
array([        inf,         inf,         inf, 3.48301086, 3.13302309,
       2.3574041 , 2.28776037, 2.28561996, 2.19061267, 2.14315083,
       2.14315083, 1.87045921, 1.82414464, 1.58763842, 1.47345133,
       1.40999223, 1.37507342, 1.35216673, 1.34507248, 1.32012389,
       1.30994891, 1.24273998, 1.15854432, 1.00072723, 0.93016386,
       0.82605321, 0.82547776, 0.68678496, 0.63088929, 0.61427436])
```

In [27]:
```
plot_imp(best_features_fscore, feature_importances_fscore, 'F-Score', 'red')
```

```
<vega.vegalite.VegaLite at 0x7fd618e16208>
```

Out[27]:



We can see that the types of ads and whether the ads have been seen are very important in predicting the revenue metrics.

In [28]:
```
from sklearn.model_selection import train_test_split
D_train, D_test, t_train, t_test = train_test_split(Data, target, test_size = 0.3, ra
```

4 Hyperparameter Tuning

4.1 K-Nearest Neighbors (KNN)

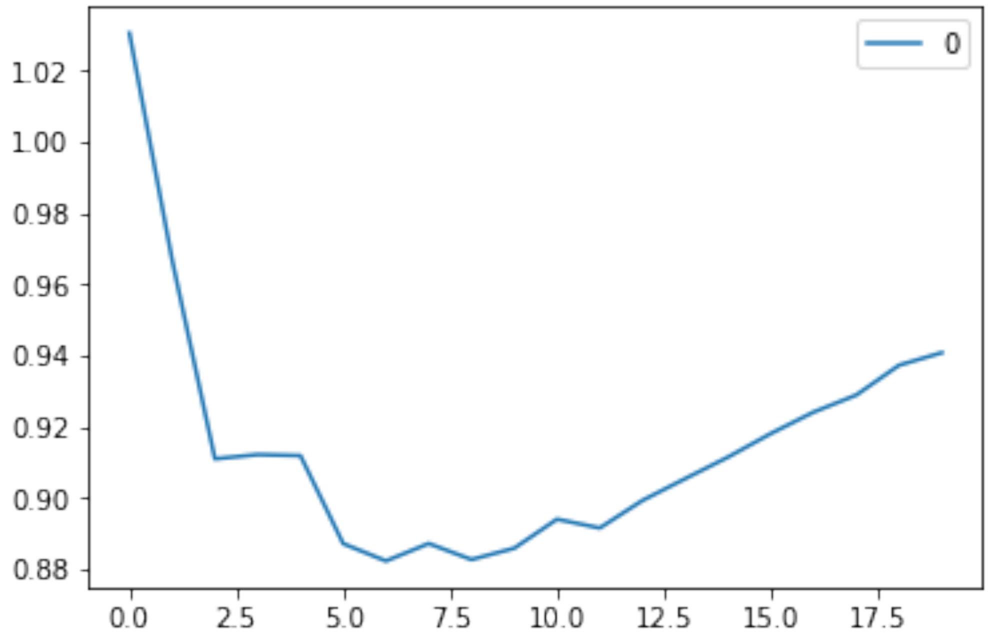
```
In [29]: #import required packages
         from sklearn import neighbors
         from sklearn.metrics import mean_squared_error
         from math import sqrt
         import matplotlib.pyplot as plt
         %matplotlib inline
```

```
In [30]: rmse_val = [] #to store rmse values for different k
         for K in range(20):
             K = K+1
             model = neighbors.KNeighborsRegressor(n_neighbors = K)

             model.fit(D_train, t_train)  #fit the model
             pred=model.predict(D_test) #make prediction on test set
             error = sqrt(mean_squared_error(t_test,pred)) #calculate rmse
             rmse_val.append(error) #store rmse values
             print('RMSE value for k= ' , K , 'is:', error)
```

```
RMSE value for k=  1 is: 1.0305901855035802
RMSE value for k=  2 is: 0.9668304318014557
RMSE value for k=  3 is: 0.9109773698428836
RMSE value for k=  4 is: 0.9121679066096606
RMSE value for k=  5 is: 0.9118821823732283
RMSE value for k=  6 is: 0.8871510359011127
RMSE value for k=  7 is: 0.8822886683302644
RMSE value for k=  8 is: 0.8871866502659109
RMSE value for k=  9 is: 0.8826859720612534
RMSE value for k=  10 is: 0.8858686419609396
RMSE value for k=  11 is: 0.8940196615518102
RMSE value for k=  12 is: 0.8915489698134217
RMSE value for k=  13 is: 0.8992975644553651
RMSE value for k=  14 is: 0.9054229256019375
RMSE value for k=  15 is: 0.9114705054417236
RMSE value for k=  16 is: 0.9180749088602547
RMSE value for k=  17 is: 0.924104784720914
RMSE value for k=  18 is: 0.9289048673367724
RMSE value for k=  19 is: 0.937285152653569
RMSE value for k=  20 is: 0.9408014764984509
```

```
In [31]: curve = pd.DataFrame(rmse_val) #elbow curve
         curve.plot()
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd60a53bfd0>
```



The smallest RMSE is: 0.8822886683302644, for k= 7 and p=2(default)

```
In [32]: rmse_val = [] #to store rmse values for different k
         for K in range(25):
             K = K+1
             model = neighbors.KNeighborsRegressor(n_neighbors = K,p=1)

             model.fit(D_train, t_train)  #fit the model
             pred=model.predict(D_test) #make prediction on test set
             error = sqrt(mean_squared_error(t_test,pred)) #calculate rmse
             rmse_val.append(error) #store rmse values
             print('RMSE value for k= ' , K , 'is:', error)
```

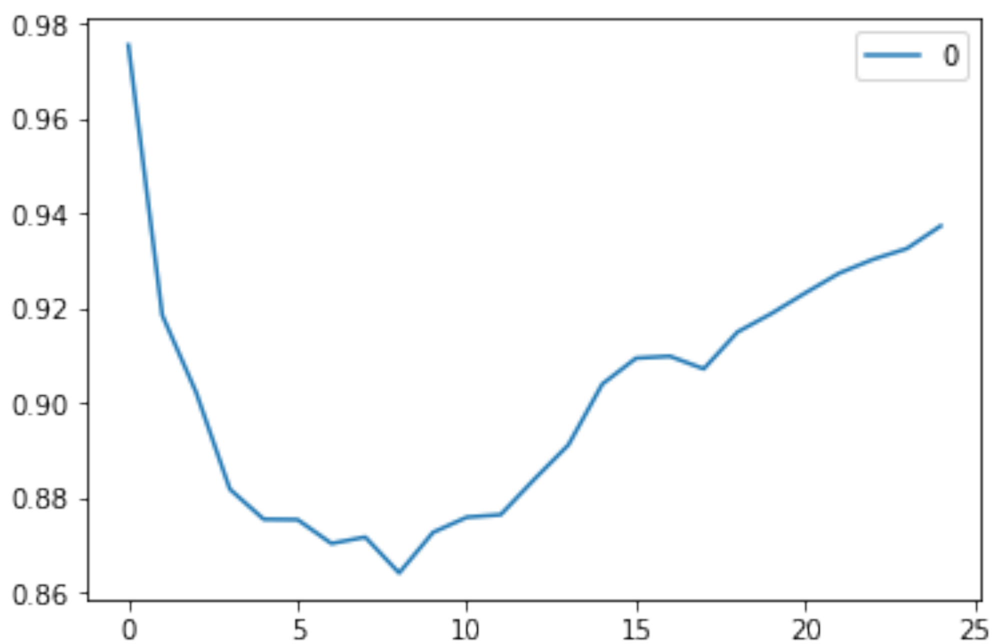
```
RMSE value for k=  1 is: 0.9754934640791116
RMSE value for k=  2 is: 0.9185379244512423
RMSE value for k=  3 is: 0.9022682251453219
RMSE value for k=  4 is: 0.8817949326745231
RMSE value for k=  5 is: 0.8754613601858245
RMSE value for k=  6 is: 0.8753859300406561
RMSE value for k=  7 is: 0.8703575851943868
RMSE value for k=  8 is: 0.8717098876309661
RMSE value for k=  9 is: 0.8641370999444125
RMSE value for k=  10 is: 0.872690511205818
RMSE value for k=  11 is: 0.8759062064030289
RMSE value for k=  12 is: 0.8764325571222461
RMSE value for k=  13 is: 0.8839443120892517
RMSE value for k=  14 is: 0.8911470222774874
```

```
RMSE value for k=  15 is: 0.9039661307492504
RMSE value for k=  16 is: 0.9094549004305397
RMSE value for k=  17 is: 0.909852218525278
RMSE value for k=  18 is: 0.907193310481234
RMSE value for k=  19 is: 0.914984749367884
RMSE value for k=  20 is: 0.9188612421621171
RMSE value for k=  21 is: 0.9232024644147652
RMSE value for k=  22 is: 0.9273640820671579
RMSE value for k=  23 is: 0.9302926275251043
RMSE value for k=  24 is: 0.9325667782737221
RMSE value for k=  25 is: 0.9373440194802309
```

The smallest RMSE is: 0.8641370999444125, for k= 9 and p=1

```
In [33]: curve = pd.DataFrame(rmse_val) #elbow curve
         curve.plot()
```

```
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd62854e5c0>
```




```
In [34]: rmse_val = [] #to store rmse values for different k
         for K in range(25):
             K = K+1
             model = neighbors.KNeighborsRegressor(n_neighbors = K,p=5)

             model.fit(D_train, t_train)  #fit the model
```

```
            pred=model.predict(D_test) #make prediction on test set
            error = sqrt(mean_squared_error(t_test,pred)) #calculate rmse
            rmse_val.append(error) #store rmse values
            print('RMSE value for k= ' , K , 'is:', error)
```

```
RMSE value for k=  1 is: 1.0499407321406684
RMSE value for k=  2 is: 1.0163218675445673
RMSE value for k=  3 is: 0.9442848418230485
RMSE value for k=  4 is: 0.9222274238337047
RMSE value for k=  5 is: 0.907214171331412
RMSE value for k=  6 is: 0.9038363509365478
RMSE value for k=  7 is: 0.8975310369715938
RMSE value for k=  8 is: 0.9125083185354782
RMSE value for k=  9 is: 0.9125300359587158
RMSE value for k=  10 is: 0.9144712152780051
RMSE value for k=  11 is: 0.9209540671220642
RMSE value for k=  12 is: 0.9185739861272637
RMSE value for k=  13 is: 0.9236131367318158
RMSE value for k=  14 is: 0.9306891647896928
RMSE value for k=  15 is: 0.9364084539188752
RMSE value for k=  16 is: 0.9440083481534253
RMSE value for k=  17 is: 0.9506324457984555
RMSE value for k=  18 is: 0.955111886450721
RMSE value for k=  19 is: 0.9610883496575272
RMSE value for k=  20 is: 0.9613974855100886
RMSE value for k=  21 is: 0.9652849667458809
RMSE value for k=  22 is: 0.9700752408619284
RMSE value for k=  23 is: 0.9701463868566038
RMSE value for k=  24 is: 0.9711833328190026
RMSE value for k=  25 is: 0.974978202191287
```

The smallest RMSE is: 0.8975310369715938, for k= 7 and p=5
In brief, the KNN model with the lowest RMES has k= 5 and p= 1.

4.2 Decision Trees (DT)

```
In [35]: from sklearn.tree import DecisionTreeRegressor
```

RMSE value for depth= 9 is: 0.7947932049414127

```
In [36]: rmse_val = [] #to store rmse values for different d
         for a in range(2,18):
                for d in range (18):
                     d = d+1
                     model = DecisionTreeRegressor(criterion='mse',max_depth = d, random_state
                     model.fit(D_train, t_train)  #fit the model
                     pred=model.predict(D_test) #make prediction on test set
                     error = sqrt(mean_squared_error(t_test,pred)) #calculate rmse
```

```
                    rmse_val.append(error) #store rmse values
                    print('RMSE value for depth= ' , d ,'min_samples_split',a, 'is:', error)
```

```
RMSE value for depth=  1 min_samples_split 2 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 2 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 2 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 2 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 2 is: 0.9408641404956216
RMSE value for depth=  6 min_samples_split 2 is: 0.9152915045454926
RMSE value for depth=  7 min_samples_split 2 is: 0.9516866737787589
RMSE value for depth=  8 min_samples_split 2 is: 0.9770450592116736
RMSE value for depth=  9 min_samples_split 2 is: 0.9580805269437394
RMSE value for depth=  10 min_samples_split 2 is: 0.9744222887519833
RMSE value for depth=  11 min_samples_split 2 is: 0.9816351975201582
RMSE value for depth=  12 min_samples_split 2 is: 1.0040313501605298
RMSE value for depth=  13 min_samples_split 2 is: 1.0114895495839782
RMSE value for depth=  14 min_samples_split 2 is: 1.018006623982051
RMSE value for depth=  15 min_samples_split 2 is: 1.0247891054127312
RMSE value for depth=  16 min_samples_split 2 is: 1.04150056032121
RMSE value for depth=  17 min_samples_split 2 is: 1.0430821044410599
RMSE value for depth=  18 min_samples_split 2 is: 1.0429207347593774
RMSE value for depth=  1 min_samples_split 3 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 3 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 3 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 3 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 3 is: 0.9408641404956216
RMSE value for depth=  6 min_samples_split 3 is: 0.9152915045454925
RMSE value for depth=  7 min_samples_split 3 is: 0.9576147948225475
RMSE value for depth=  8 min_samples_split 3 is: 0.9544403761602701
RMSE value for depth=  9 min_samples_split 3 is: 0.9409769784231523
RMSE value for depth=  10 min_samples_split 3 is: 0.9900494273063439
RMSE value for depth=  11 min_samples_split 3 is: 0.9682818809397267
RMSE value for depth=  12 min_samples_split 3 is: 1.0082795618940112
RMSE value for depth=  13 min_samples_split 3 is: 0.997764013167296
RMSE value for depth=  14 min_samples_split 3 is: 1.0308912485161121
RMSE value for depth=  15 min_samples_split 3 is: 1.0253967297909923
RMSE value for depth=  16 min_samples_split 3 is: 1.0003196948939264
RMSE value for depth=  17 min_samples_split 3 is: 1.0264668974585256
RMSE value for depth=  18 min_samples_split 3 is: 0.94735140906888
RMSE value for depth=  1 min_samples_split 4 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 4 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 4 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 4 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 4 is: 0.9408641404956216
RMSE value for depth=  6 min_samples_split 4 is: 0.9139234750123398
RMSE value for depth=  7 min_samples_split 4 is: 0.9721978505451312
RMSE value for depth=  8 min_samples_split 4 is: 0.937562282974085
RMSE value for depth=  9 min_samples_split 4 is: 0.9524594841006592
```

```
RMSE value for depth=  10 min_samples_split 4 is: 0.9803654218176874
RMSE value for depth=  11 min_samples_split 4 is: 0.9555810699329209
RMSE value for depth=  12 min_samples_split 4 is: 0.9690268777712994
RMSE value for depth=  13 min_samples_split 4 is: 0.9755214765894623
RMSE value for depth=  14 min_samples_split 4 is: 1.0313667754932283
RMSE value for depth=  15 min_samples_split 4 is: 1.0127929505632076
RMSE value for depth=  16 min_samples_split 4 is: 0.9777399928333098
RMSE value for depth=  17 min_samples_split 4 is: 0.9849165319257369
RMSE value for depth=  18 min_samples_split 4 is: 0.9794774676173978
RMSE value for depth=  1 min_samples_split 5 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 5 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 5 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 5 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 5 is: 0.9408641404956216
RMSE value for depth=  6 min_samples_split 5 is: 0.9139234750123398
RMSE value for depth=  7 min_samples_split 5 is: 0.9565426490409666
RMSE value for depth=  8 min_samples_split 5 is: 0.9499561908759547
RMSE value for depth=  9 min_samples_split 5 is: 0.9720561794605885
RMSE value for depth=  10 min_samples_split 5 is: 0.9621421996942324
RMSE value for depth=  11 min_samples_split 5 is: 0.9722760285595674
RMSE value for depth=  12 min_samples_split 5 is: 0.956388747708171
RMSE value for depth=  13 min_samples_split 5 is: 0.9383670618904962
RMSE value for depth=  14 min_samples_split 5 is: 1.0092276945193324
RMSE value for depth=  15 min_samples_split 5 is: 0.9927252582874991
RMSE value for depth=  16 min_samples_split 5 is: 1.0290753805699389
RMSE value for depth=  17 min_samples_split 5 is: 1.001344482353621
RMSE value for depth=  18 min_samples_split 5 is: 0.9484472207457079
RMSE value for depth=  1 min_samples_split 6 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 6 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 6 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 6 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 6 is: 0.9457993661460047
RMSE value for depth=  6 min_samples_split 6 is: 0.9192815624016106
RMSE value for depth=  7 min_samples_split 6 is: 0.9501831754419329
RMSE value for depth=  8 min_samples_split 6 is: 0.9621853202637888
RMSE value for depth=  9 min_samples_split 6 is: 0.9808414743092657
RMSE value for depth=  10 min_samples_split 6 is: 0.9363762061104458
RMSE value for depth=  11 min_samples_split 6 is: 0.9385008115652329
RMSE value for depth=  12 min_samples_split 6 is: 0.9591336890234762
RMSE value for depth=  13 min_samples_split 6 is: 0.9774089892051793
RMSE value for depth=  14 min_samples_split 6 is: 0.9873882537165444
RMSE value for depth=  15 min_samples_split 6 is: 0.9557745828697536
RMSE value for depth=  16 min_samples_split 6 is: 0.9732661624094049
RMSE value for depth=  17 min_samples_split 6 is: 0.9553155755921062
RMSE value for depth=  18 min_samples_split 6 is: 1.0024877113103443
RMSE value for depth=  1 min_samples_split 7 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 7 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 7 is: 0.9487521096441158
```

```
RMSE value for depth=  4 min_samples_split 7 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 7 is: 0.9457993661460047
RMSE value for depth=  6 min_samples_split 7 is: 0.9192815624016106
RMSE value for depth=  7 min_samples_split 7 is: 0.9175667942206585
RMSE value for depth=  8 min_samples_split 7 is: 0.9122952041859508
RMSE value for depth=  9 min_samples_split 7 is: 0.9131264468157961
RMSE value for depth=  10 min_samples_split 7 is: 0.9168818116928621
RMSE value for depth=  11 min_samples_split 7 is: 0.8995450524814689
RMSE value for depth=  12 min_samples_split 7 is: 0.9195464462374009
RMSE value for depth=  13 min_samples_split 7 is: 0.9282497546506864
RMSE value for depth=  14 min_samples_split 7 is: 0.9570953221381386
RMSE value for depth=  15 min_samples_split 7 is: 0.9343078145606696
RMSE value for depth=  16 min_samples_split 7 is: 0.9365613361789235
RMSE value for depth=  17 min_samples_split 7 is: 0.9349813740548878
RMSE value for depth=  18 min_samples_split 7 is: 0.9432139167321331
RMSE value for depth=  1 min_samples_split 8 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 8 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 8 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 8 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 8 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 8 is: 0.9189214441151194
RMSE value for depth=  7 min_samples_split 8 is: 0.9248533660889076
RMSE value for depth=  8 min_samples_split 8 is: 0.9121855822631736
RMSE value for depth=  9 min_samples_split 8 is: 0.9298509186247598
RMSE value for depth=  10 min_samples_split 8 is: 0.8998782073356418
RMSE value for depth=  11 min_samples_split 8 is: 0.9230906307708848
RMSE value for depth=  12 min_samples_split 8 is: 0.9171231308766206
RMSE value for depth=  13 min_samples_split 8 is: 0.9433083661874856
RMSE value for depth=  14 min_samples_split 8 is: 0.9533714439467617
RMSE value for depth=  15 min_samples_split 8 is: 0.9229605086225603
RMSE value for depth=  16 min_samples_split 8 is: 0.9432683017626776
RMSE value for depth=  17 min_samples_split 8 is: 0.9295708860070286
RMSE value for depth=  18 min_samples_split 8 is: 0.9276956916251823
RMSE value for depth=  1 min_samples_split 9 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 9 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 9 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 9 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 9 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 9 is: 0.9189214441151194
RMSE value for depth=  7 min_samples_split 9 is: 0.9247400883595502
RMSE value for depth=  8 min_samples_split 9 is: 0.9124346443584093
RMSE value for depth=  9 min_samples_split 9 is: 0.8959240160964399
RMSE value for depth=  10 min_samples_split 9 is: 0.9178070931143089
RMSE value for depth=  11 min_samples_split 9 is: 0.9081801052233127
RMSE value for depth=  12 min_samples_split 9 is: 0.9066511513802845
RMSE value for depth=  13 min_samples_split 9 is: 0.9327131803813008
RMSE value for depth=  14 min_samples_split 9 is: 0.939421750190192
RMSE value for depth=  15 min_samples_split 9 is: 0.9211101178693303
```

```
RMSE value for depth=  16 min_samples_split 9 is: 0.9407430186516973
RMSE value for depth=  17 min_samples_split 9 is: 0.9423138603601051
RMSE value for depth=  18 min_samples_split 9 is: 0.9084467293736442
RMSE value for depth=  1 min_samples_split 10 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 10 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 10 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 10 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 10 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 10 is: 0.9007992054582288
RMSE value for depth=  7 min_samples_split 10 is: 0.9064264948938796
RMSE value for depth=  8 min_samples_split 10 is: 0.8938689724181236
RMSE value for depth=  9 min_samples_split 10 is: 0.8858508955583547
RMSE value for depth=  10 min_samples_split 10 is: 0.9030116062810293
RMSE value for depth=  11 min_samples_split 10 is: 0.8972984454434886
RMSE value for depth=  12 min_samples_split 10 is: 0.8889969815962467
RMSE value for depth=  13 min_samples_split 10 is: 0.9133524596841347
RMSE value for depth=  14 min_samples_split 10 is: 0.9162493972529643
RMSE value for depth=  15 min_samples_split 10 is: 0.8969636512376982
RMSE value for depth=  16 min_samples_split 10 is: 0.8926068288458031
RMSE value for depth=  17 min_samples_split 10 is: 0.9017285502880767
RMSE value for depth=  18 min_samples_split 10 is: 0.9188063960825295
RMSE value for depth=  1 min_samples_split 11 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 11 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 11 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 11 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 11 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 11 is: 0.9007992054582288
RMSE value for depth=  7 min_samples_split 11 is: 0.9064264948938796
RMSE value for depth=  8 min_samples_split 11 is: 0.8932951551826669
RMSE value for depth=  9 min_samples_split 11 is: 0.8771508555845494
RMSE value for depth=  10 min_samples_split 11 is: 0.8733443391059006
RMSE value for depth=  11 min_samples_split 11 is: 0.8952242287621783
RMSE value for depth=  12 min_samples_split 11 is: 0.8911448159340548
RMSE value for depth=  13 min_samples_split 11 is: 0.8901931574672417
RMSE value for depth=  14 min_samples_split 11 is: 0.8856207695559332
RMSE value for depth=  15 min_samples_split 11 is: 0.9204382039535467
RMSE value for depth=  16 min_samples_split 11 is: 0.8947780124491386
RMSE value for depth=  17 min_samples_split 11 is: 0.9161451044009172
RMSE value for depth=  18 min_samples_split 11 is: 0.9208309321285107
RMSE value for depth=  1 min_samples_split 12 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 12 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 12 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 12 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 12 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 12 is: 0.9007992054582288
RMSE value for depth=  7 min_samples_split 12 is: 0.899213167307035
RMSE value for depth=  8 min_samples_split 12 is: 0.8981497929246609
RMSE value for depth=  9 min_samples_split 12 is: 0.9102498763913568
```

```
RMSE value for depth=  10 min_samples_split 12 is: 0.9033182748429504
RMSE value for depth=  11 min_samples_split 12 is: 0.9013890053982786
RMSE value for depth=  12 min_samples_split 12 is: 0.9171841043930317
RMSE value for depth=  13 min_samples_split 12 is: 0.8928874325219394
RMSE value for depth=  14 min_samples_split 12 is: 0.9192349184388399
RMSE value for depth=  15 min_samples_split 12 is: 0.8949422389159847
RMSE value for depth=  16 min_samples_split 12 is: 0.9193042329276155
RMSE value for depth=  17 min_samples_split 12 is: 0.9202232555475587
RMSE value for depth=  18 min_samples_split 12 is: 0.9028922687636509
RMSE value for depth=  1 min_samples_split 13 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 13 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 13 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 13 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 13 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 13 is: 0.8992091034697172
RMSE value for depth=  7 min_samples_split 13 is: 0.8976354192345822
RMSE value for depth=  8 min_samples_split 13 is: 0.8962730787151211
RMSE value for depth=  9 min_samples_split 13 is: 0.9092033066539105
RMSE value for depth=  10 min_samples_split 13 is: 0.9012505067687364
RMSE value for depth=  11 min_samples_split 13 is: 0.8777429843033014
RMSE value for depth=  12 min_samples_split 13 is: 0.9065303986994313
RMSE value for depth=  13 min_samples_split 13 is: 0.9166114994213893
RMSE value for depth=  14 min_samples_split 13 is: 0.9176322636090691
RMSE value for depth=  15 min_samples_split 13 is: 0.9012602158744577
RMSE value for depth=  16 min_samples_split 13 is: 0.9181102549821702
RMSE value for depth=  17 min_samples_split 13 is: 0.8963905371790084
RMSE value for depth=  18 min_samples_split 13 is: 0.9180123194953939
RMSE value for depth=  1 min_samples_split 14 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 14 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 14 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 14 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 14 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 14 is: 0.8992091034697172
RMSE value for depth=  7 min_samples_split 14 is: 0.8976354192345822
RMSE value for depth=  8 min_samples_split 14 is: 0.8944729976038739
RMSE value for depth=  9 min_samples_split 14 is: 0.8893604959513479
RMSE value for depth=  10 min_samples_split 14 is: 0.8782465238117382
RMSE value for depth=  11 min_samples_split 14 is: 0.8982544730282493
RMSE value for depth=  12 min_samples_split 14 is: 0.9094633658648539
RMSE value for depth=  13 min_samples_split 14 is: 0.8955767184950529
RMSE value for depth=  14 min_samples_split 14 is: 0.8968504222759625
RMSE value for depth=  15 min_samples_split 14 is: 0.8971750976188841
RMSE value for depth=  16 min_samples_split 14 is: 0.901008562761951
RMSE value for depth=  17 min_samples_split 14 is: 0.9182089015498854
RMSE value for depth=  18 min_samples_split 14 is: 0.9007655888382615
RMSE value for depth=  1 min_samples_split 15 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 15 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 15 is: 0.9487521096441158
```

```
RMSE value for depth=  4 min_samples_split 15 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 15 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 15 is: 0.9217133851177084
RMSE value for depth=  7 min_samples_split 15 is: 0.9153393945588957
RMSE value for depth=  8 min_samples_split 15 is: 0.9122383486838631
RMSE value for depth=  9 min_samples_split 15 is: 0.9268373061927166
RMSE value for depth=  10 min_samples_split 15 is: 0.8963749330926662
RMSE value for depth=  11 min_samples_split 15 is: 0.8962807705066533
RMSE value for depth=  12 min_samples_split 15 is: 0.93328080412891
RMSE value for depth=  13 min_samples_split 15 is: 0.914215380357223
RMSE value for depth=  14 min_samples_split 15 is: 0.9168034730606994
RMSE value for depth=  15 min_samples_split 15 is: 0.9163624090629485
RMSE value for depth=  16 min_samples_split 15 is: 0.9364883964119359
RMSE value for depth=  17 min_samples_split 15 is: 0.9417907538778796
RMSE value for depth=  18 min_samples_split 15 is: 0.9173712708231994
RMSE value for depth=  1 min_samples_split 16 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 16 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 16 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 16 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 16 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 16 is: 0.9217133851177084
RMSE value for depth=  7 min_samples_split 16 is: 0.9153393945588957
RMSE value for depth=  8 min_samples_split 16 is: 0.9120779571115968
RMSE value for depth=  9 min_samples_split 16 is: 0.9067792876383953
RMSE value for depth=  10 min_samples_split 16 is: 0.9164611094579432
RMSE value for depth=  11 min_samples_split 16 is: 0.9169027159426308
RMSE value for depth=  12 min_samples_split 16 is: 0.9073720939551871
RMSE value for depth=  13 min_samples_split 16 is: 0.9131148181872543
RMSE value for depth=  14 min_samples_split 16 is: 0.9150508862131177
RMSE value for depth=  15 min_samples_split 16 is: 0.9186480700418079
RMSE value for depth=  16 min_samples_split 16 is: 0.9183908826785241
RMSE value for depth=  17 min_samples_split 16 is: 0.9151828254147532
RMSE value for depth=  18 min_samples_split 16 is: 0.9361300929547759
RMSE value for depth=  1 min_samples_split 17 is: 1.0525647787918906
RMSE value for depth=  2 min_samples_split 17 is: 0.9734807658161001
RMSE value for depth=  3 min_samples_split 17 is: 0.9487521096441158
RMSE value for depth=  4 min_samples_split 17 is: 0.9458230716508771
RMSE value for depth=  5 min_samples_split 17 is: 0.9454493484503833
RMSE value for depth=  6 min_samples_split 17 is: 0.9217133851177084
RMSE value for depth=  7 min_samples_split 17 is: 0.9153393945588957
RMSE value for depth=  8 min_samples_split 17 is: 0.9139040762486507
RMSE value for depth=  9 min_samples_split 17 is: 0.9084858959705936
RMSE value for depth=  10 min_samples_split 17 is: 0.8976056343730219
RMSE value for depth=  11 min_samples_split 17 is: 0.9186189523744953
RMSE value for depth=  12 min_samples_split 17 is: 0.9089793236175675
RMSE value for depth=  13 min_samples_split 17 is: 0.9122917637432316
RMSE value for depth=  14 min_samples_split 17 is: 0.9373761614408859
RMSE value for depth=  15 min_samples_split 17 is: 0.9376262878270469
```

```
RMSE value for depth=  16 min_samples_split 17 is: 0.9374272131506893
RMSE value for depth=  17 min_samples_split 17 is: 0.9346975980377512
RMSE value for depth=  18 min_samples_split 17 is: 0.9349134919120116
```

```
In [37]: min(rmse_val)
```

```
Out[37]: 0.8733443391059006
```

The DT regressor with the smallest RMSE of 0.8733443391059006, has value for depth of 10 min_samples_split of 11

4.3 Bayian Regression and Other Linear model

```
In [38]: from sklearn.naive_bayes import GaussianNB
         from sklearn import linear_model
         from sklearn import svm
```

```
In [39]: classifiers = [

             linear_model.BayesianRidge(),
             linear_model.ARDRegression(),
             linear_model.PassiveAggressiveRegressor(),
             linear_model.TheilSenRegressor(),
             linear_model.Lasso(),
             linear_model.LinearRegression()]
```

```
In [40]: for item in classifiers:

             clf = item
             clf.fit(D_train, t_train)
             pred=clf.predict(D_test)
             error = sqrt(mean_squared_error(t_test,pred)) #calculate rmse
             print( error)
```

```
0.8681414823811188
0.8683883021598259
1.1912748059788325
0.8741054590895775
1.1512085865490016
0.8680385761153382
```

The linear model with the lowest RMSE is Bayesian Ridge with the value of 0.8681414823811188, however, ARD Regression did not perform much worst with the RMSE of 0.8683883021598259.

5 Performance Comparison

Overall, most of models have the RMSE is around 1 dollar. After parameter tuning, the best performance belongs to Bayesian Ridge followed by ARD Regression. DT regressor performed worse than KNN regressor in this case.

6 Limitations and Proposed Solutions

The modelling process has come with a few flaws and limitations. First, we only used a smaller portion of the data for training and testing; this may have introduced some biases in to the models.

In addition, we also ignored a few features such as countryId based on their correlations with the target variable; we may have missed information created by the intereactions of these features with other features, which may influence the target variable.

The process of features selections and hyper-parameter tuning is still primitive; a more in-depth analysis and parameter search may improve the performance of the models; a Grid-search and feature-selection pipeline is a good example, we have encountered some technical issue trying to use the pipeline, therefore, it was not included in this report.

The ARDRegression out-performed other regression, however, we have not consider any parameters tuning for this model; an inclusive parameter tuning process may improve the performance of this model.

Lastly, we used RMSE as our metric of comparison. The conclusions made from this metric alone is not necessarily conclusive.

7 Summary

In summary, the analysis managed to use a sample of the data to consider a few models. In this case, the linear regession seems to outperform DT and KNN.In addition, we have learned that out self-generated features provided some insights; for example, the types of the ads are important in the process of predicting the revenue- metrics. Further analysis with the full scale data should be considered before deployment in case.

8 References

- Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830.
- Aksakalli, V. (2019). MATH2319: Machine Learning, week 1-12, lecture notes.