**Predict CLTV of a customer**  
  
Can you help the company identify the most potential customers?

**Problem Statement**

VahanBima is one of the leading insurance companies in India. It provides motor **vehicle insurances** at best prices with 24/7 claim settlement.  It offers different types of policies for both personal and commercial vehicles. It has established its brand across different regions in India.   
  
Around 90% of the businesses today use personalized services. The company wants to launch different personalized experience programs for customers of VahanBima. The personalized experience can be dedicated resources for claim settlement, different kinds of services at doorstep, etc. In order to do so, they would like to **segment the customers into different tiers** based on their customer lifetime value (CLTV).  
  
In order to do it, they would like to predict the customer lifetime value based on the activity and interaction of the customer with the platform. So, as a part of this challenge, your task at hand is to build a high performance and interpretable machine learning model to predict the CLTV based on the user and policy data.  
  
  
**About the Dataset**  
  
You are provided with the sample dataset of the company holding the information of customers and policy such as highest qualification of the user, total income earned by a customer in a year, employee status,  policy opted by the user, type of policy and so on and the target variable indicating the total customer lifetime value.  
  
  
**Data Dictionary**  
  
You are provided with 3 files - train.csv, test.csv and sample\_submission.csv  
  
**Train Set**  
  
You are provided with around 90K records containing the attributes of the user and policy and the target variable *cltv* indicating the total customer lifetime value.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| id | Unique identifier of a customer |
| gender | Gender of the customer |
| area | Area of the customer |
| qualification | Highest Qualification of the customer |
| income | Income earned in a year (in rupees) |
| marital\_status | Marital Status of the customer {0:Single, 1: Married} |
| vintage | No. of years since the first policy date |
| claim\_amount | Total Amount Claimed by the customer (in rupees) |
| num\_policies | Total no. of policies issued by the customer |
| policy | Active policy of the customer |
| type\_of\_policy | Type of active policy |
| *cltv* | *Customer life time value (Target Variable)* |

**Test Set**  
  
You are provided with around 60K records containing only the attributes of the user and policy and you need to predict the target variable *cltv* indicating the total customer lifetime value.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| id | Unique identifier of a customer |
| gender | Gender of the customer |
| area | Area of the user |
| qualification | Highest Qualification of the customer |
| income | Income earned in a year (in rupees) |
| marital\_status | Marital Status of the customer {0:Single, 1: Married} |
| Vintage | No. of years since the first policy date |
| claim\_amount | Total Amount Claimed by the customer (in rupees) |
| num\_policies | Total no. of policies issued by the customer |
| policy | Active policy of the customer |
| type\_of\_policy | Type of active policy |

**Submission File Format**  
  
The solution file must contain the format similar to that of sample submission.**sample\_submission.csv**contains 2 variables - id and cltv. 

|  |  |
| --- | --- |
| **Variable** | **Description** |
| id | Unique identifier of a customer |
| cltv | Customer life time value |

**Evaluation metric**  
  
The evaluation metric for this hackathon would be r2\_score.  
  
  
**Public and Private Split**  
  
Test data is further divided into Public (40%) and Private (60%) data.   
  
Your initial responses will be checked and scored on the Public data. The final rankings would be based on your private score which will be published once the competition is over.  
  
  
**Submission Tutorials**

1. All Submissions are to be done at the solution checker tab.
2. For a step by step view on how to make a submission check the below video

**Feature Engineering :**

1. **Correlation:**

correlation = data['claim\_amount'].corr(data['cltv'])

print("Correlation between var1 and var2: ", correlation)

**output:**

Correlation between var1 and var2: 0.18034399105720555

1. **Anova test**

import pandas as pd

from scipy.stats import f\_oneway

# load your data into a dataframe

df = pd.read\_csv('train\_BRCpofr.csv')

categorical\_variables=[]

for col in data.columns:

    if data[col].dtype == 'object':

        categorical\_variables.append(col)

# list of categorical variables

# categorical\_variables = ["cat\_var1", "cat\_var2", "cat\_var3", "cat\_var4", "cat\_var5", "cat\_var6", "cat\_var7", "cat\_var8", "cat\_var9", "cat\_var10"]

features\_cltv=[]

# continuous variable

continuous\_variable = 'cltv'

categorical\_variables.append('marital\_status')

categorical\_variables.append('vintage')

# Perform ANOVA tests

for cat\_var in categorical\_variables:

    # check the count of each level in the categorical variable

    counts = df[cat\_var].value\_counts()

    # select only the levels that have enough data

    selected\_levels = counts[counts > 2].index

    # pass selected levels to the anova test

    f\_statistic, p\_value = f\_oneway(\*[df[df[cat\_var] == level][continuous\_variable].values for level in selected\_levels])

    if p\_value<=0.05:

        features\_cltv.append(cat\_var)

    print("ANOVA results for " + cat\_var + ":")

    print("F-statistic: ", f\_statistic)

    print("P-value: ", p\_value)

    print("\n")

print("Important feature with ref to ANOVA test are  : ",features\_cltv)

**output:**

ANOVA results for gender:

F-statistic: 0.1552119399388717

P-value: 0.693604522475604

ANOVA results for area:

F-statistic: 1612.3403505017125

P-value: 0.0

ANOVA results for qualification:

F-statistic: 98.99276677439613

P-value: 1.1363927139220727e-43

ANOVA results for income:

F-statistic: 186.77051836371825

P-value: 9.665251183824116e-121

ANOVA results for num\_policies:

F-statistic: 13345.072198127686

P-value: 0.0

ANOVA results for policy:

F-statistic: 171.43391165436717

P-value: 4.893535365473623e-75

ANOVA results for type\_of\_policy:

F-statistic: 50.686293268116536

P-value: 9.992979900289914e-23

ANOVA results for marital\_status:

F-statistic: 543.568751832195

P-value: 7.207849478628778e-120

ANOVA results for vintage:

F-statistic: 20.48763481150897

P-value: 2.618741516428911e-31

Important feature with ref to ANOVA test are : ['area', 'qualification', 'income', 'num\_policies', 'policy', 'type\_of\_policy', 'marital\_status', 'vintage']

-----------------------------------------------------------------------------------------------------------------------------

**Main Problem pipeline:**

import pandas as pd

# load the training dataset

data = pd.read\_csv('train\_BRCpofr.csv')

data.head()

# Split the data into features and target

import numpy as np

X = data.drop(['id','cltv'], axis=1)

y = data['cltv']

# Train the model

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.linear\_model import LinearRegression,Lasso,Ridge

from sklearn.svm import SVR

from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor

import xgboost as xg

import numpy as np

# Define preprocessing for numeric columns (scale them)

numeric\_features = [5,6]

numeric\_transformer = Pipeline(steps=[

    ('scaler', StandardScaler())])

# Define preprocessing for categorical features (encode them)

categorical\_features = [0,1,2,3,4,7,8,9]

categorical\_transformer = Pipeline(steps=[

    ('onehot', OneHotEncoder(handle\_unknown='ignore'))])

# Combine preprocessing steps

preprocessor = ColumnTransformer(

    transformers=[

        ('num', numeric\_transformer, numeric\_features),

        ('cat', categorical\_transformer, categorical\_features)])

# Create preprocessing and training pipeline

pipeline = Pipeline(steps=[('preprocessor', preprocessor),

                           ('regressor', GradientBoostingRegressor(n\_estimators=85,max\_depth=3,learning\_rate=0.075))])

from sklearn.model\_selection import KFold

# Define the number of splits and the random seed

kf = KFold(n\_splits=5, random\_state=42, shuffle=True)

for fold,(train\_index, test\_index) in enumerate(kf.split(X)):

    X\_train, X\_test = X.iloc[train\_index], X.iloc[test\_index]

    y\_train, y\_test = y.iloc[train\_index], y.iloc[test\_index]

    print(X\_train.shape, X\_test.shape)

    print (f"model\_{fold}")

    model = pipeline.fit(X\_train, (y\_train))

    # Get predictions

    import matplotlib.pyplot as plt

    import seaborn as sns

    %matplotlib inline

    from sklearn.metrics import mean\_squared\_error, r2\_score

    predictions = model.predict(X\_test)

    # Display metrics

    mse = mean\_squared\_error(y\_test, predictions)

    print("MSE:", mse)

    rmse = np.sqrt(mse)

    print("RMSE:", rmse)

    r2 = r2\_score(y\_test, predictions)

    print("R2:", r2)

    # Plot predicted vs actual

    plt.scatter(y\_test, predictions)

    plt.xlabel('Actual Labels')

    plt.ylabel('Predicted Labels')

    plt.title('cltv Predictions')

    z = np.polyfit(y\_test, predictions, 1)

    p = np.poly1d(z)

    plt.plot(y\_test,p(y\_test), color='magenta')

    plt.show()

    import pandas as pd

    # train = pd.read\_csv('train.csv')

    test  = pd.read\_csv("test\_koRSKBP.csv")

    predictions = model.predict(test.drop(['id'],axis=1))

    # predictions = model.predict(test)

    # predictions = scaler\_y.inverse\_transform(predictions)

    # #taking the mean of cltv from the training set

    # test['cltv']=train['cltv'].mean()

    test\_id=test['id']

    df = pd.DataFrame({'id':np.array(test\_id),'cltv':predictions}, columns = ['id','cltv'])

    #creating the sample submission file

    sample\_submission = df[['id','cltv']]

    sample\_submission.to\_csv(f"sample\_submission\_GBR\_model\_{fold}.csv",index=False)

    print("\*"\*50)

    print("\n\n")

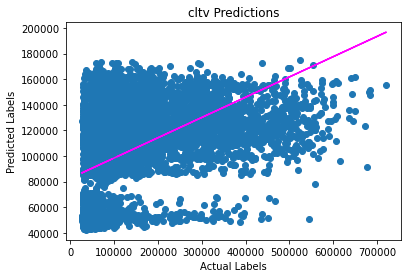
(71513, 10) (17879, 10)

model\_0

MSE: 6727247126.980326

RMSE: 82019.79716495478

R2: 0.1592906042286829



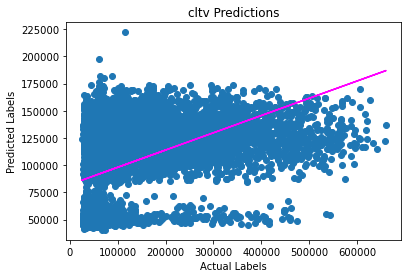
(71513, 10) (17879, 10)

model\_1

MSE: 6786994172.140711

RMSE: 82383.21535446835

R2: 0.15955534413072148



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

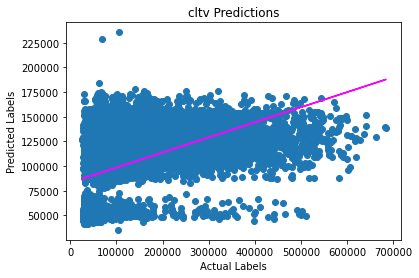
(71514, 10) (17878, 10)

model\_2

MSE: 7103068609.453619

RMSE: 84279.70461180805

R2: 0.1564860177586661



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

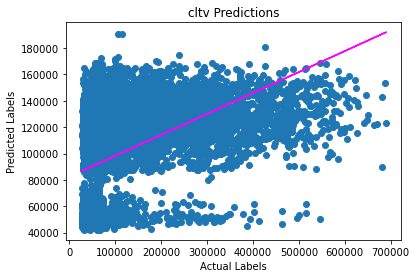
**(71514, 10) (17878, 10)**

**model\_3**

**MSE: 6867654547.163231**

**RMSE: 82871.31317387961**

**R2: 0.1648486456580801**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

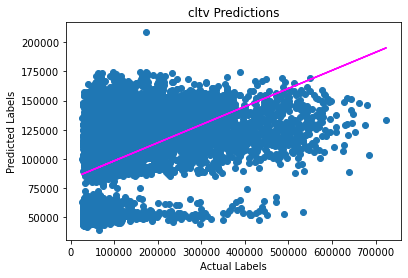
(71514, 10) (17878, 10)

model\_4

MSE: 7006212908.317569

RMSE: 83703.12364731419

R2: 0.1589795492341205



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*