In [1]:

```
import numpy as np
import pandas as pd
import sklearn
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

In [2]:

```
X_train = pd.read_csv('./Data/orange_small_train.data', sep='\t')
X_test = pd.read_csv('./Data/orange_small_test.data', sep='\t')
y_train_churn = pd.read_csv('./Data/orange_small_train_churn.labels.txt',header=None)
y_train_apt = pd.read_csv('./Data/orange_small_train_appetency.labels',header=None)
y_train_upsell = pd.read_csv('./Data/orange_small_train_upselling.labels',header=None)
```

In [3]:

```
y_train_apt.value_counts(normalize=True), y_train_churn.value_counts(normalize=True), y_tra
    normalize=True)
```

Out[3]:

```
(-1    0.9822
  1    0.0178
 dtype: float64,
 -1    0.92656
  1    0.07344
 dtype: float64,
 -1    0.92636
  1    0.07364
 dtype: float64)
```

In [4]:

```
# dropping any column with missing value

# collect minimum no of records to be present value for data
# using data with high percentage of missing records makes bad training set and will induce
missing_perc = 20
min_count = int(((100 - missing_perc) / 100) * X_train.shape[0] + 1)
min_count
```

Out[4]:

```
40001
```

Feature engineering

In [5]:

```
#drop all missing and only continous variable
prefered_uniques = 10
# create a list of numeric values and less than prefered uniques categorical columns
fe_unique_columns = list(X_train.select_dtypes(include='number').columns) + list(
    (X_train.select_dtypes(include='object').nunique() < prefered_uniques).index[
        X_train.select_dtypes(include='object').nunique() < prefered_uniques])
# collect only numeric features
X_train_all_missing_drop_continous = X_train.dropna( axis=1,thresh=min_count).dropna(axis=0
numeric_col_after_drop = X_train_all_missing_drop_continous.columns
# create X_train with categorical variables with less than 10 uniques
X_train_all_missing_dropped_prefered_unique_cat = X_train[fe_unique_columns].dropna( axis=1
feature_prefered_columns = list(X_train_all_missing_dropped_prefered_unique_cat.columns)
# creating dummy variables for the data with  categorical variables
X_train_encoded = pd.get_dummies(X_train_all_missing_dropped_prefered_unique_cat,drop_first
X_train_numeric = X_train_all_missing_drop_continous
del X_train_all_missing_drop_continous
# selecting X_test with the appropriate predictors for numeric predictors and selected uniq
X_test_numeric = X_test[numeric_col_after_drop]
X_test_encoded = pd.get_dummies(X_test[feature_prefered_columns],drop_first=True)[X_train_e
```

In [6]:

```
X_train_encoded.columns
```

Out[6]:

```
Index(['Var6', 'Var7', 'Var13', 'Var21', 'Var22', 'Var24', 'Var25', 'Var28',
       'Var35', 'Var38', 'Var44', 'Var57', 'Var65', 'Var73', 'Var74', 'Var7
6',
       'Var78', 'Var81', 'Var83', 'Var85', 'Var109', 'Var112', 'Var113',
       'Var119', 'Var123', 'Var125', 'Var132', 'Var133', 'Var134', 'Var140',
       'Var143', 'Var144', 'Var149', 'Var153', 'Var160', 'Var163', 'Var173',
       'Var181', 'Var196_JA1C', 'Var196_mKeq', 'Var196_z3mO', 'Var203_F3hy',
       'Var203_HLqf', 'Var203_dgxZ', 'Var205_VpdQ', 'Var205_sJzTlal',
       'Var208_sBgB', 'Var210_7A3j', 'Var210_DM_V', 'Var210_g5HH',
       'Var210_uKAI', 'Var211_Mtgm', 'Var218_cJvF', 'Var221_JIiEFBU',
       'Var221_QKW8DRm', 'Var221_d0EEeJi', 'Var221_oslk', 'Var221_z4pH',
       'Var221_zCkv', 'Var223_M_8D', 'Var223_bCPvVye', 'Var223_jySVZNlOJy',
       'Var227_6fzt', 'Var227_RAYp', 'Var227_ZI9m', 'Var227_nIGXDli',
       'Var227_nIGjgSB', 'Var227_vJ_w8kB'],
      dtype='object')
```

In [7]:

```
features=np.array(X_train_encoded.columns[:38])
```

In [8]:

```
type(features)
```

Out[8]:

```
numpy.ndarray
```

In [9]:

```
extra_features=['Var196','Var203','Var205','Var208','Var210','Var211','Var218','Var221','Va
```

In [10]:

```
#features.append(extra_features)
for i in range(len(extra_features)):
    features=np.append(features,extra_features[i])
```

In [11]:

```
features
```

Out[11]:

```
array(['Var6', 'Var7', 'Var13', 'Var21', 'Var22', 'Var24', 'Var25',
       'Var28', 'Var35', 'Var38', 'Var44', 'Var57', 'Var65', 'Var73',
       'Var74', 'Var76', 'Var78', 'Var81', 'Var83', 'Var85', 'Var109',
       'Var112', 'Var113', 'Var119', 'Var123', 'Var125', 'Var132',
       'Var133', 'Var134', 'Var140', 'Var143', 'Var144', 'Var149',
       'Var153', 'Var160', 'Var163', 'Var173', 'Var181', 'Var196',
       'Var203', 'Var205', 'Var208', 'Var210', 'Var211', 'Var218',
       'Var221', 'Var223', 'Var227'], dtype=object)
```

In [12]:

```
X_train = pd.read_csv('./Data/orange_small_train.data', sep='\t')
X_test = pd.read_csv('./Data/orange_small_test.data', sep='\t')
y_train_churn = pd.read_csv('./Data/orange_small_train_churn.labels.txt',header=None)
y_train_apt = pd.read_csv('./Data/orange_small_train_appetency.labels',header=None)
y_train_upsell = pd.read_csv('./Data/orange_small_train_upselling.labels',header=None)
```

In [13]:

```
X_train=X_train[features]
X_test=X_test[features]
```

In [14]:

```
X_train.columns
```

Out[14]:

```
Index(['Var6', 'Var7', 'Var13', 'Var21', 'Var22', 'Var24', 'Var25', 'Var28',
       'Var35', 'Var38', 'Var44', 'Var57', 'Var65', 'Var73', 'Var74', 'Var7
6',
       'Var78', 'Var81', 'Var83', 'Var85', 'Var109', 'Var112', 'Var113',
       'Var119', 'Var123', 'Var125', 'Var132', 'Var133', 'Var134', 'Var140',
       'Var143', 'Var144', 'Var149', 'Var153', 'Var160', 'Var163', 'Var173',
       'Var181', 'Var196', 'Var203', 'Var205', 'Var208', 'Var210', 'Var211',
       'Var218', 'Var221', 'Var223', 'Var227'],
      dtype='object')
```

In [15]:

```
X_train=pd.get_dummies(X_train,columns=extra_features)
X_test=pd.get_dummies(X_test,columns=extra_features)
```

In [16]:

```
X_train['churn']=y_train_churn
X_train['apt']=y_train_apt
X_train['upsell']=y_train_upsell
```

In [17]:

```
X_train.columns
```

Out[17]:

```
Index(['Var6', 'Var7', 'Var13', 'Var21', 'Var22', 'Var24', 'Var25', 'Var28',
       'Var35', 'Var38', 'Var44', 'Var57', 'Var65', 'Var73', 'Var74', 'Var7
6',
       'Var78', 'Var81', 'Var83', 'Var85', 'Var109', 'Var112', 'Var113',
       'Var119', 'Var123', 'Var125', 'Var132', 'Var133', 'Var134', 'Var140',
       'Var143', 'Var144', 'Var149', 'Var153', 'Var160', 'Var163', 'Var173',
       'Var181', 'Var196_1K8T', 'Var196_JA1C', 'Var196_mKeq', 'Var196_z3mO',
       'Var203_9_Y1', 'Var203_F3hy', 'Var203_HLqf', 'Var203_dgxZ',
       'Var203_pybr', 'Var205_09_Q', 'Var205_VpdQ', 'Var205_sJzTlal',
       'Var208_kIsH', 'Var208_sBgB', 'Var210_3av_', 'Var210_7A3j',
       'Var210_DM_V', 'Var210_g5HH', 'Var210_oT7d', 'Var210_uKAI',
       'Var211_L84s', 'Var211_Mtgm', 'Var218_UYBR', 'Var218_cJvF',
       'Var221_Al6ZaUT', 'Var221_JIiEFBU', 'Var221_QKW8DRm', 'Var221_d0EEeJ
i',
       'Var221_oslk', 'Var221_z4pH', 'Var221_zCkv', 'Var223_LM8l689qOp',
       'Var223_M_8D', 'Var223_bCPvVye', 'Var223_jySVZNlOJy', 'Var227_02N6s8
f',
       'Var227_6fzt', 'Var227_RAYp', 'Var227_ZI9m', 'Var227_nIGXDli',
       'Var227_nIGjgSB', 'Var227_vJ_w8kB', 'churn', 'apt', 'upsell'],
      dtype='object')
```

In [18]:

```
X_train.dropna(inplace=True)
X_test.dropna(inplace=True)
```

In [19]:

```
X_train.shape
```

Out[19]:

```
(42153, 83)
```

In [20]:

```
X_train.head()
```

Out[20]:

	Var6	Var7	Var13	Var21	Var22	Var24	Var25	Var28	Var35	Var38	...	Var227_02I
0	1526.0	7.0	184.0	464.0	580.0	14.0	128.0	166.56	0.0	3570.0	...	
1	525.0	0.0	0.0	168.0	210.0	2.0	24.0	353.52	0.0	4764966.0	...	
2	5236.0	7.0	904.0	1212.0	1515.0	26.0	816.0	220.08	0.0	5883894.0	...	
4	1029.0	7.0	3216.0	64.0	80.0	4.0	64.0	200.00	0.0	0.0	...	
5	658.0	7.0	3156.0	224.0	280.0	2.0	72.0	200.00	5.0	0.0	...	

5 rows × 83 columns

In [21]:

```
xc_train,xc_test=train_test_split(X_train,train_size=0.9)
```

In [22]:

```
xc_train.columns
```

Out[22]:

```
Index(['Var6', 'Var7', 'Var13', 'Var21', 'Var22', 'Var24', 'Var25', 'Var28',
       'Var35', 'Var38', 'Var44', 'Var57', 'Var65', 'Var73', 'Var74', 'Var7
6',
       'Var78', 'Var81', 'Var83', 'Var85', 'Var109', 'Var112', 'Var113',
       'Var119', 'Var123', 'Var125', 'Var132', 'Var133', 'Var134', 'Var140',
       'Var143', 'Var144', 'Var149', 'Var153', 'Var160', 'Var163', 'Var173',
       'Var181', 'Var196_1K8T', 'Var196_JA1C', 'Var196_mKeq', 'Var196_z3mO',
       'Var203_9_Y1', 'Var203_F3hy', 'Var203_HLqf', 'Var203_dgxZ',
       'Var203_pybr', 'Var205_09_Q', 'Var205_VpdQ', 'Var205_sJzTlal',
       'Var208_kIsH', 'Var208_sBgB', 'Var210_3av_', 'Var210_7A3j',
       'Var210_DM_V', 'Var210_g5HH', 'Var210_oT7d', 'Var210_uKAI',
       'Var211_L84s', 'Var211_Mtgm', 'Var218_UYBR', 'Var218_cJvF',
       'Var221_Al6ZaUT', 'Var221_JIiEFBU', 'Var221_QKW8DRm', 'Var221_d0EEeJ
i',
       'Var221_oslk', 'Var221_z4pH', 'Var221_zCkv', 'Var223_LM8l689qOp',
       'Var223_M_8D', 'Var223_bCPvVye', 'Var223_jySVZNlOJy', 'Var227_02N6s8
f',
       'Var227_6fzt', 'Var227_RAYp', 'Var227_ZI9m', 'Var227_nIGXDli',
       'Var227_nIGjgSB', 'Var227_vJ_w8kB', 'churn', 'apt', 'upsell'],
      dtype='object')
```

In [23]:

```
xc_test.head()
```

Out[23]:

/ar35	Var38	...	Var227_02N6s8f	Var227_6fzt	Var227_RAYp	Var227_Zl9m	Var227_nIGXDli	Va
0.0	210516.0	...	0	1	0	0	0	
0.0	240900.0	...	0	0	0	1	0	
0.0	4954734.0	...	0	0	1	0	0	
0.0	3114666.0	...	0	0	0	0	1	
0.0	587202.0	...	0	0	1	0	0	

In [24]:

```
max_depths=[5,10,15,20,25,30]
```

In [26]:

```
yc_train,yc_test=xc_train['churn'],xc_test['churn']
yu_train,yu_test=xc_train['upsell'],xc_test['upsell']
ya_train,ya_test=xc_train['apt'],xc_test['apt']
```

In [27]:

```
xc_train,xc_test=xc_train.drop(columns=['apt','churn','upsell']),xc_test.drop(columns=['apt
```

In [32]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [33]:

```
accuracy_churn=[]
```

In [35]:

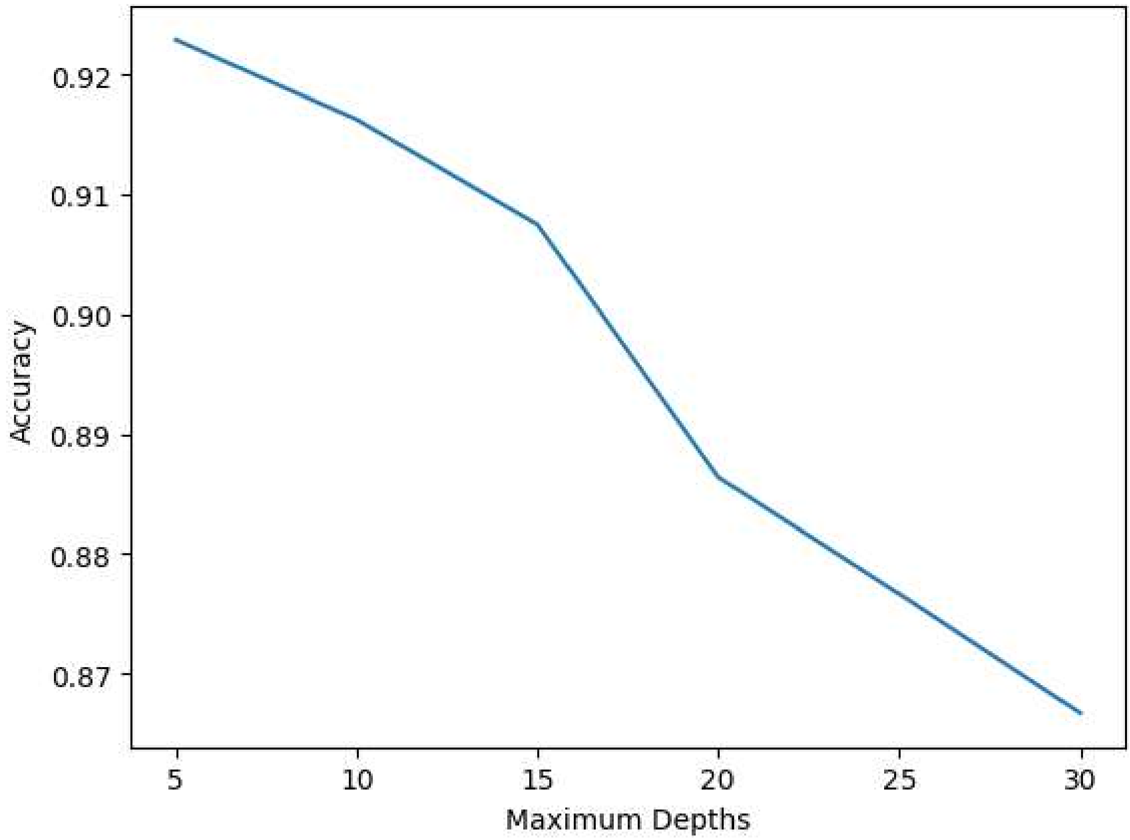
```
#for churn

for i in range(len(max_depths)):
    modelchurn=DecisionTreeClassifier(max_depth=max_depths[i])

    modelchurn.fit(xc_train,yc_train)

    preds=modelchurn.predict(xc_test)

    accuracy=accuracy_score(yc_test,preds)

    accuracy_churn.append(accuracy)
```

In [37]:

```
plt.plot(max_depths,accuracy_churn)

plt.xlabel('Maximum Depths')
plt.ylabel('Accuracy')

plt.plot()
```

Out[37]:

```
[]
```



In [38]:

```
accuracy_upsell=[]
```

In [40]:

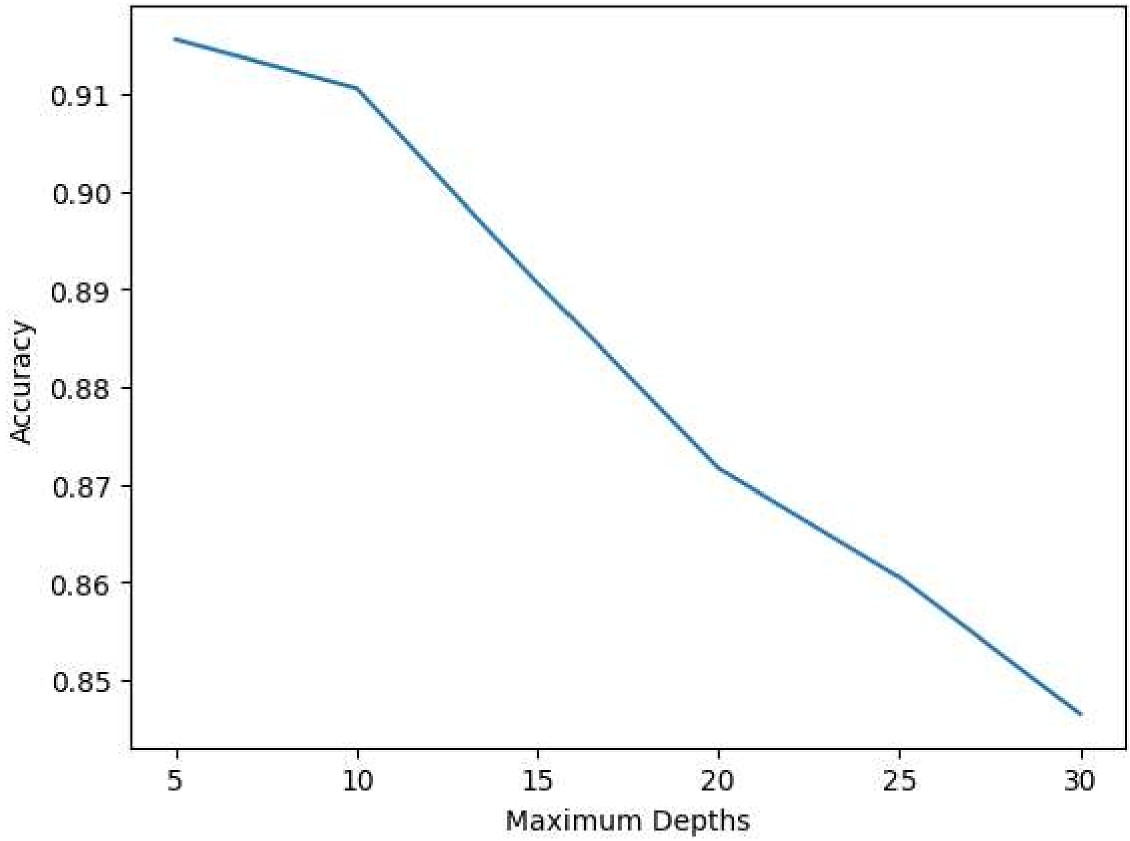
```
#for upsell

for i in range(len(max_depths)):
    modelchurn=DecisionTreeClassifier(max_depth=max_depths[i])

    modelchurn.fit(xc_train,yu_train)

    preds=modelchurn.predict(xc_test)

    accuracy=accuracy_score(yu_test,preds)

    accuracy_upsell.append(accuracy)
```

In [41]:

```
plt.plot(max_depths,accuracy_upsell)

plt.xlabel('Maximum Depths')
plt.ylabel('Accuracy')

plt.plot()
```

Out[41]:

```
[]
```



In [42]:

```
accuracy_apt=[]
```

In [44]:

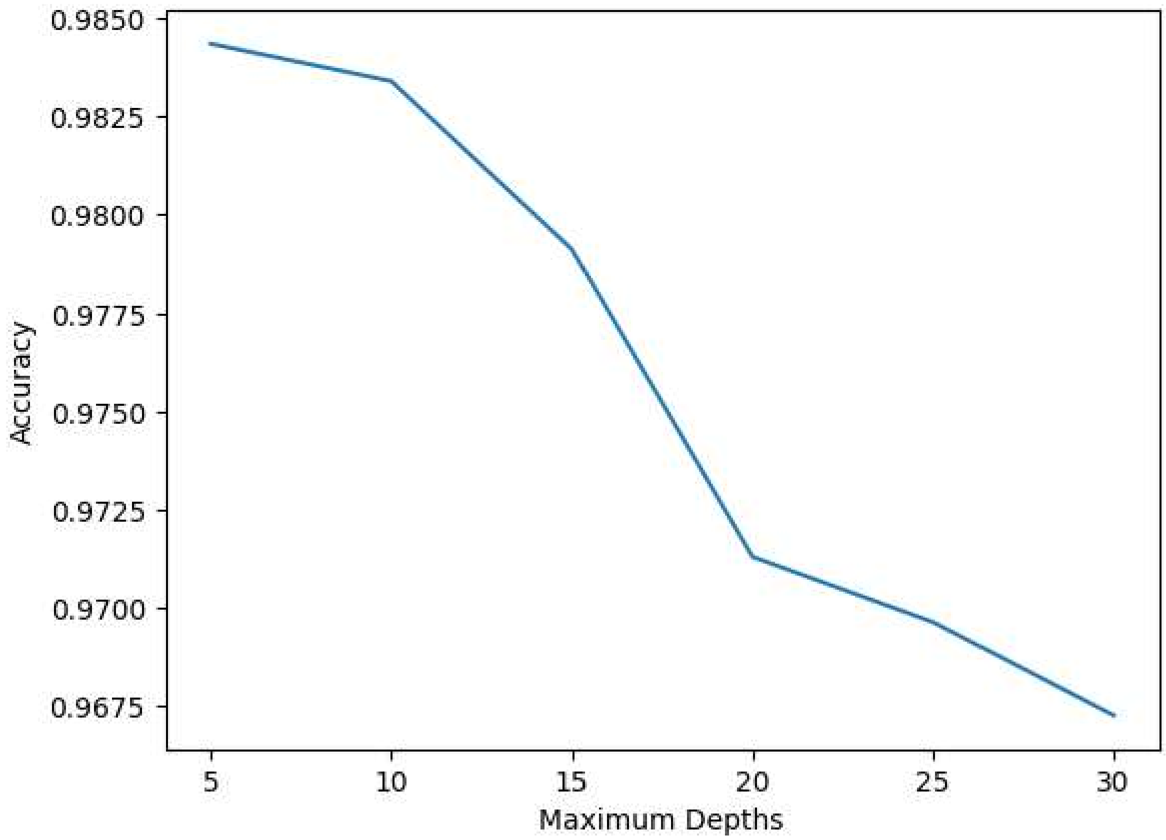
```
#for apt

for i in range(len(max_depths)):
    modelchurn=DecisionTreeClassifier(max_depth=max_depths[i])

    modelchurn.fit(xc_train,ya_train)

    preds=modelchurn.predict(xc_test)

    accuracy=accuracy_score(ya_test,preds)

    accuracy_apt.append(accuracy)
```

In [46]:

```
plt.plot(max_depths,accuracy_apt)

plt.xlabel('Maximum Depths')
plt.ylabel('Accuracy')

plt.plot()
```

Out[46]:

```
[]
```



In [47]:

```
#Therefore max_depth=5 seems suitable
```

In [48]:

```
model_churn=DecisionTreeClassifier(max_depth=5)
model_apt=DecisionTreeClassifier(max_depth=5)
model_upsell=DecisionTreeClassifier(max_depth=5)

model_churn.fit(xc_train,yc_train)
model_apt.fit(xc_train,ya_train)
model_upsell.fit(xc_train,yu_train)
```

Out[48]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

In [49]:

```
model_churn.feature_importances_
```

Out[49]:

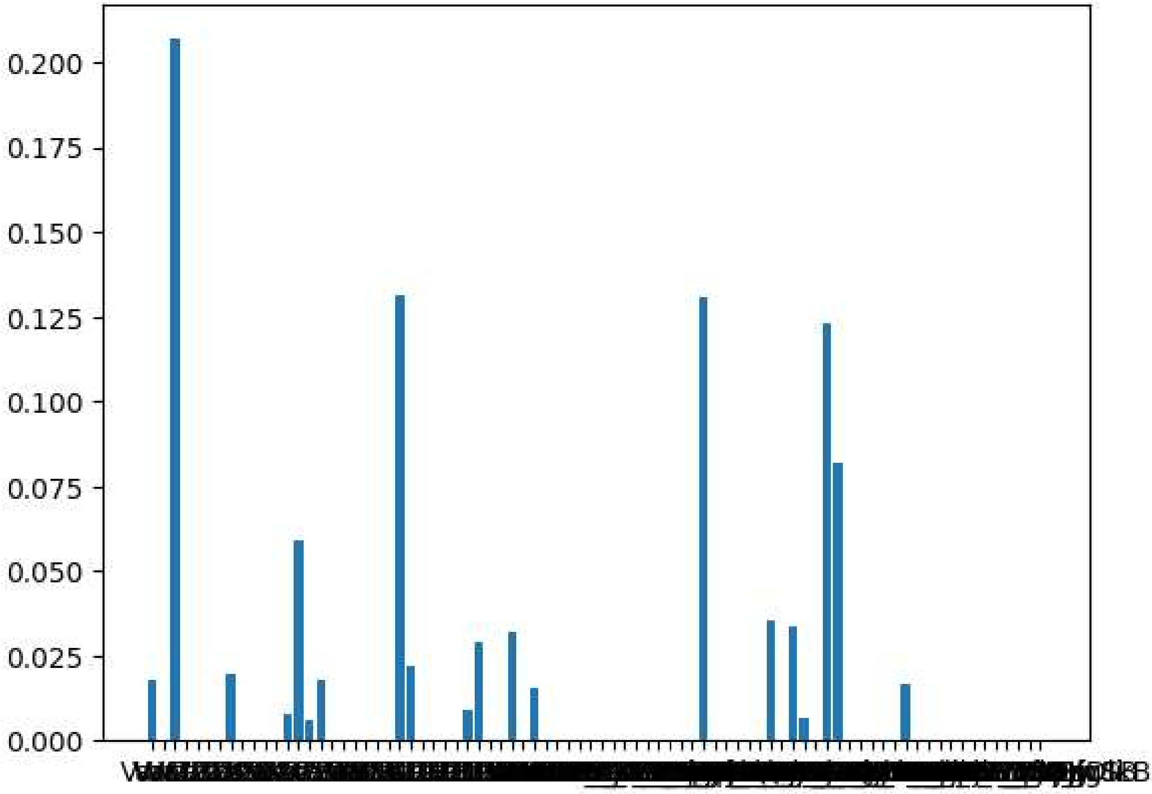
```
array([0.01765469, 0.        , 0.20698696, 0.        , 0.        ,
       0.        , 0.        , 0.01941337, 0.        , 0.        ,
       0.        , 0.        , 0.00749139, 0.05907929, 0.00625282,
       0.01759149, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.13141547, 0.02196302, 0.        ,
       0.        , 0.        , 0.        , 0.00873278, 0.02864451,
       0.        , 0.        , 0.03154154, 0.        , 0.01559011,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.13064787,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.03510297, 0.        , 0.03373739, 0.00643362, 0.        ,
       0.12297843, 0.08204916, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.01669313, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ])
```

In [50]:

```
plt.bar(xc_train.columns,model_churn.feature_importances_)

plt.plot()
```

Out[50]:

[]

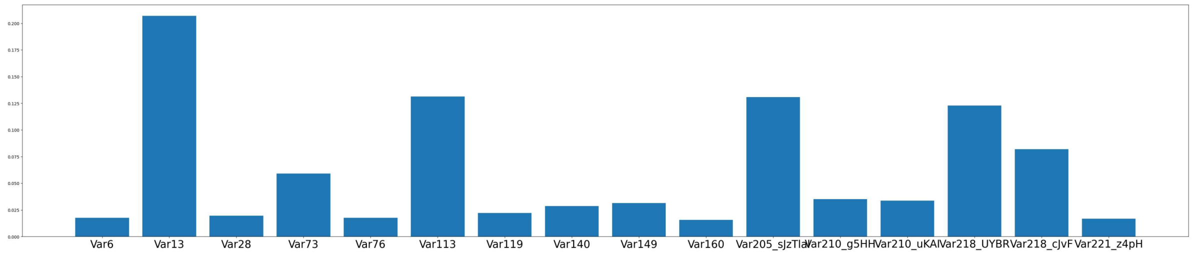


In [64]:

```
importances=model_churn.feature_importances_
indicesc=[]

for i in range(len(importances)):
    if(importances[i]>0.01):
        indicesc.append(i)
```

In [73]:

```
plt.figure(figsize=(50,10))

plt.bar(xc_train.columns[indicesc],importances[indicesc])
plt.xticks(fontsize=25)

plt.show()
```



In [74]:

```
predictions=model_churn.predict(xc_test)
```

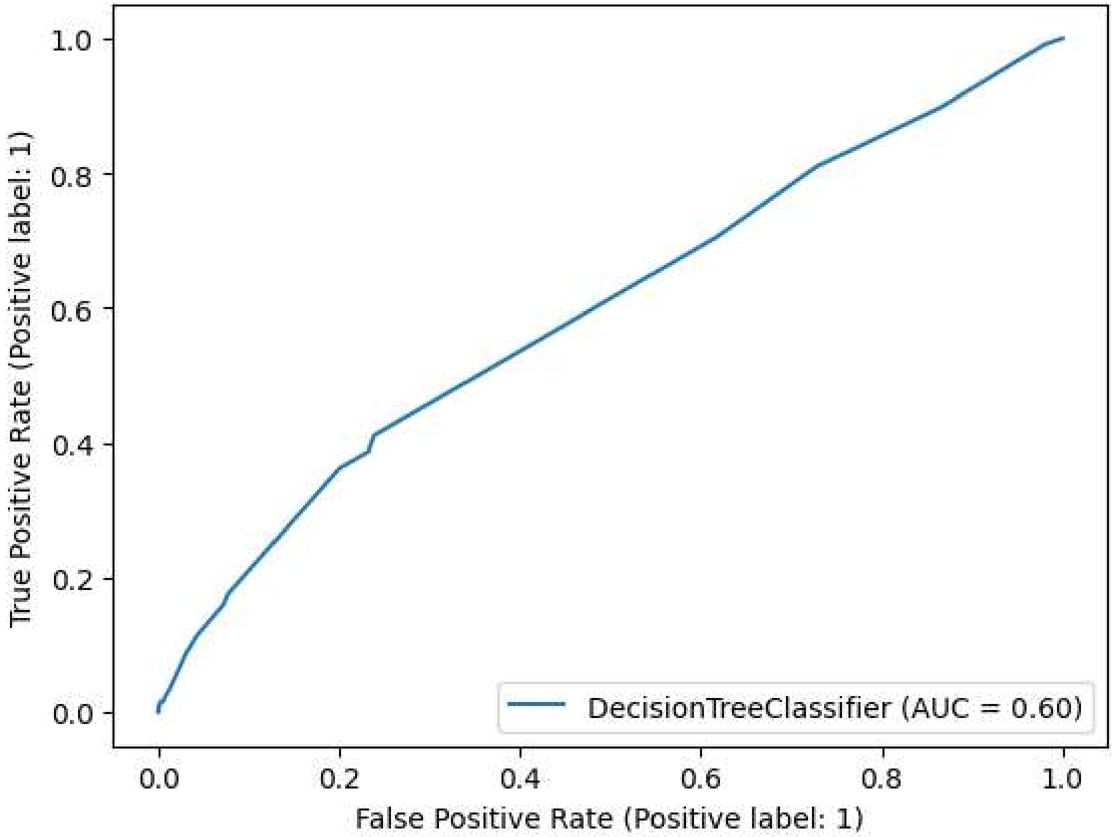
In [79]:

```
from sklearn.metrics import plot_roc_curve

plot_roc_curve(model_churn,xc_test,yc_test)

plt.plot()
```

```
C:\Users\chinm\AppData\Local\Programs\Python\Python310\lib\site-packages\skl
earn\utils\deprecation.py:87: FutureWarning: Function plot_roc_curve is depr
ecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and will be rem
oved in 1.2. Use one of the class methods: :meth:`sklearn.metrics.RocCurveDi
splay.from_predictions` or :meth:`sklearn.metrics.RocCurveDisplay.from_estim
ator`.
  warnings.warn(msg, category=FutureWarning)
```

Out[79]:

```
[]
```

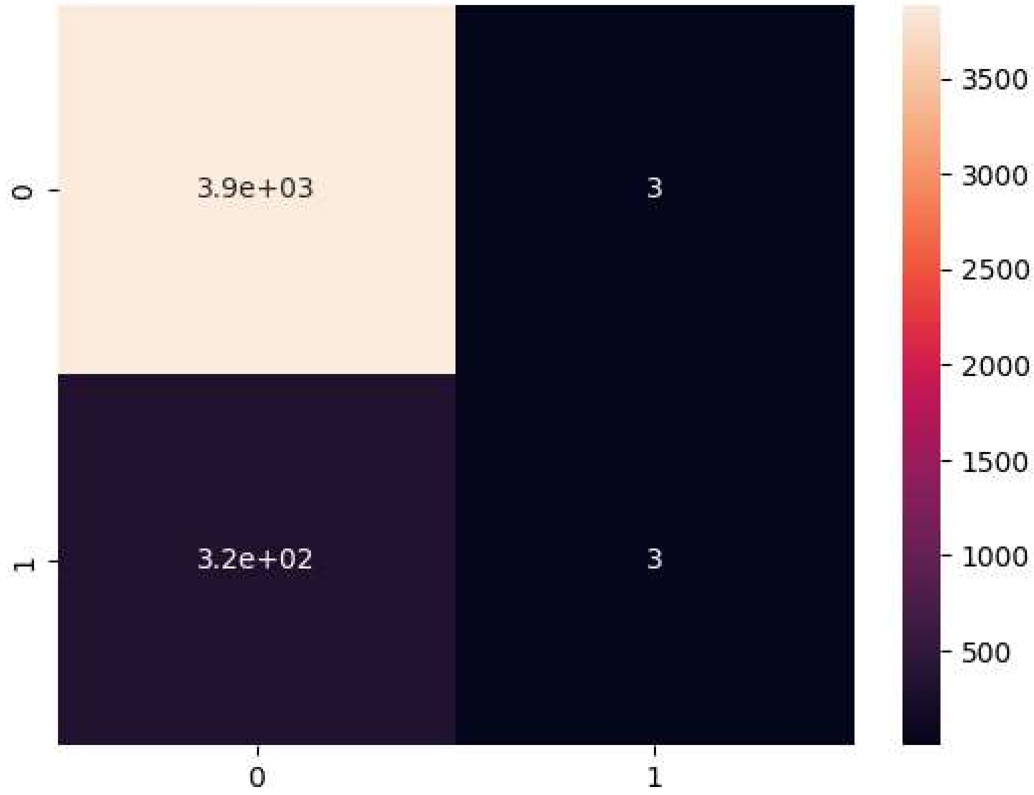


In [81]:

```
from sklearn.metrics import confusion_matrix

confusionmatrix=confusion_matrix(yc_test,predictions)

import seaborn as sns

sns.heatmap(confusionmatrix,annot=True)
```

Out[81]:

```
<AxesSubplot: >
```

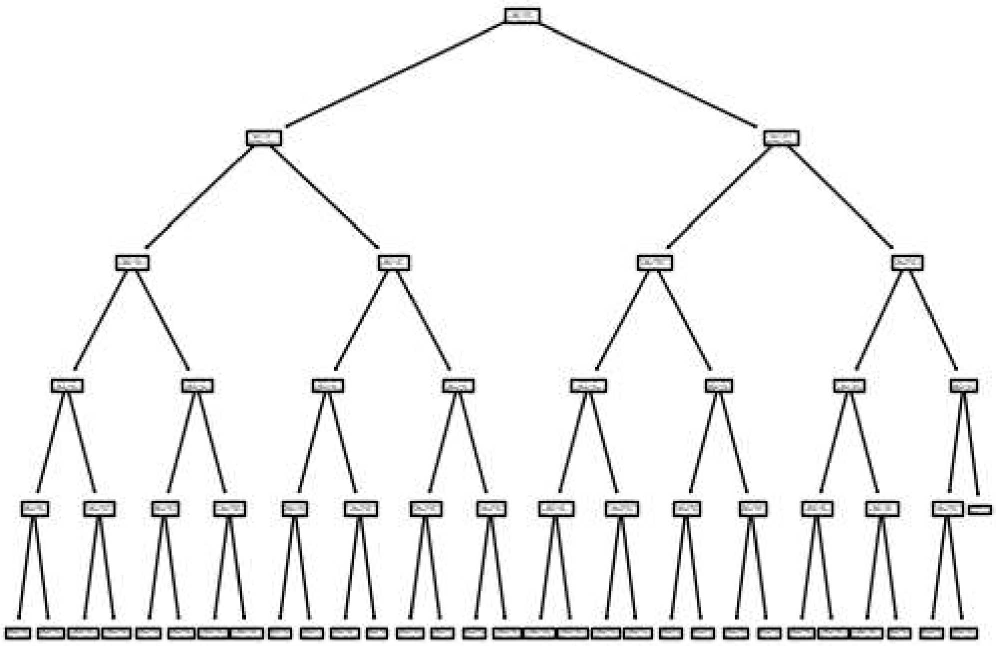


In [85]:

```
from sklearn.tree import plot_tree

plot_tree(model_churn)

plt.plot()
```

Out[85]:

```
[]
```

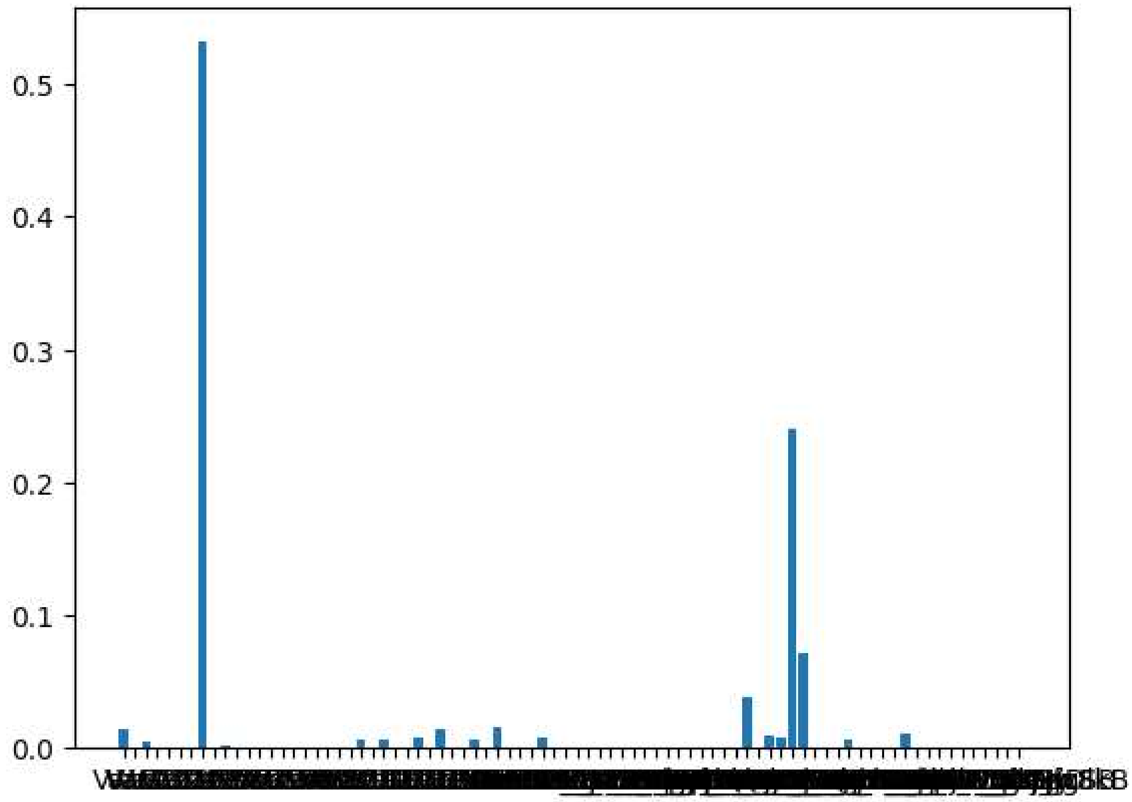


In [51]:

```
plt.bar(xc_train.columns,model_upsell.feature_importances_)

plt.plot()
```

Out[51]:

```
[]
```

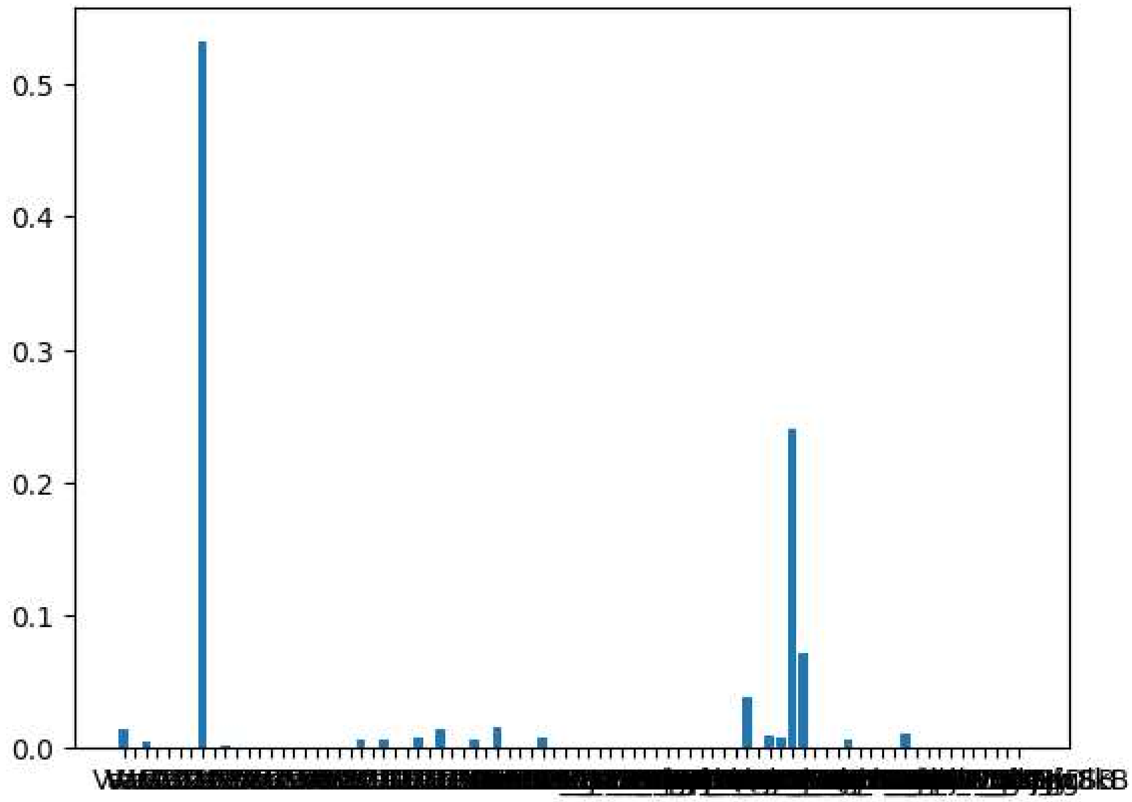


In [52]:

```
plt.bar(xc_train.columns,model_upsell.feature_importances_)

plt.plot()
```

Out[52]:

```
[]
```



In [53]:

```
X_test = pd.read_csv('./Data/orange_small_test.data', sep='\t')
```

In [54]:

```
X_test=X_test[features]
```

In [55]:

```
X_test.dropna(inplace=True)
```

In [56]:

```
X_test=pd.get_dummies(X_test,columns=extra_features)
```

In [57]:

```
X_test.head()
```

Out[57]:

	Var6	Var7	Var13	Var21	Var22	Var24	Var25	Var28	Var35	Var38	...	Var223_M_
0	1225.0	7.0	100.0	156.0	195.0	0.0	72.0	166.56	0.0	4259232.0	...	
1	259.0	0.0	0.0	192.0	240.0	0.0	40.0	300.32	5.0	4859550.0	...	
2	861.0	14.0	236.0	32.0	40.0	0.0	8.0	186.64	0.0	10038840.0	...	
3	1568.0	7.0	1232.0	448.0	560.0	4.0	88.0	166.56	0.0	116760.0	...	
4	1197.0	7.0	204.0	100.0	125.0	8.0	40.0	133.12	0.0	257772.0	...	

5 rows × 79 columns

In [58]:

```
PREDICTIONS_CHURN=model_churn.predict(X_test)
PREDICTIONS_UPSELL=model_upsell.predict(X_test)
PREDICTIONS_APT=model_apt.predict(X_test)
```

```
C:\Users\chinm\AppData\Local\Programs\Python\Python310\lib\site-packages\skl
earn\base.py:493: FutureWarning: The feature names should match those that w
ere passed during fit. Starting version 1.2, an error will be raised.
Feature names unseen at fit time:
- Var210_eyPI
Feature names seen at fit time, yet now missing:
- Var203_pybr
- Var210_oT7d

  warnings.warn(message, FutureWarning)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [58], line 1
----> 1 PREDICTIONS_CHURN=model_churn.predict(X_test)
      2 PREDICTIONS_UPSELL=model_upsell.predict(X_test)
      3 PREDICTIONS_APT=model_apt.predict(X_test)

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\tre
e\_classes.py:505, in BaseDecisionTree.predict(self, X, check_input)
    482 """Predict class or regression value for X.
    483 
    484 For a classification model, the predicted class for each sample in X
is
   (...)
    502     The predicted classes, or the predict values.
    503 """
    504 check_is_fitted(self)
--> 505 X = self._validate_X_predict(X, check_input)
    506 proba = self.tree_.predict(X)
    507 n_samples = X.shape[0]

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\tre
e\_classes.py:471, in BaseDecisionTree._validate_X_predict(self, X, check_in
put)
    469 """Validate the training data on predict (probabilities)."""
    470 if check_input:
--> 471     X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", res
et=False)
    472     if issparse(X) and (
    473         X.indices.dtype != np.intc or X.indptr.dtype != np.intc
    474     ):
    475         raise ValueError("No support for np.int64 index based sparse
matrices")

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\bas
e.py:600, in BaseEstimator._validate_data(self, X, y, reset, validate_separa
tely, **check_params)
    597     out = X, y
    599 if not no_val_X and check_params.get("ensure_2d", True):
--> 600     self._check_n_features(X, reset=reset)
    602 return out

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\bas
e.py:400, in BaseEstimator._check_n_features(self, X, reset)
```

```
    397     return
    399 if n_features != self.n_features_in_:
--> 400     raise ValueError(
    401         f"X has {n_features} features, but {self.__class__.__name__}
"
    402         f"is expecting {self.n_features_in_} features as input."
    403     )

ValueError: X has 79 features, but DecisionTreeClassifier is expecting 80 fe
atures as input.
```

In [59]:

```
csdv=np.zeros(shape=len(X_test))
```

In [60]:

```
X_test['Var203_pybr']=csdv
X_test['Var210_oT7d']=csdv
```

In [61]:

```
X_test=X_test.drop(columns=['Var210_eyPI'])
```

In [62]:

```
X_test.head()
```

Out[62]:

	Var6	Var7	Var13	Var21	Var22	Var24	Var25	Var28	Var35	Var38	...	Var223_jyS
0	1225.0	7.0	100.0	156.0	195.0	0.0	72.0	166.56	0.0	4259232.0	...	
1	259.0	0.0	0.0	192.0	240.0	0.0	40.0	300.32	5.0	4859550.0	...	
2	861.0	14.0	236.0	32.0	40.0	0.0	8.0	186.64	0.0	10038840.0	...	
3	1568.0	7.0	1232.0	448.0	560.0	4.0	88.0	166.56	0.0	116760.0	...	
4	1197.0	7.0	204.0	100.0	125.0	8.0	40.0	133.12	0.0	257772.0	...	

5 rows × 80 columns

In [63]:

```
PREDICTIONS_CHURN=model_churn.predict(X_test)
PREDICTIONS_UPSELL=model_upsell.predict(X_test)
PREDICTIONS_APT=model_apt.predict(X_test)
```

```
C:\Users\chinm\AppData\Local\Programs\Python\Python310\lib\site-packages\skl
earn\base.py:493: FutureWarning: The feature names should match those that w
ere passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
C:\Users\chinm\AppData\Local\Programs\Python\Python310\lib\site-packages\skl
earn\base.py:493: FutureWarning: The feature names should match those that w
ere passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
C:\Users\chinm\AppData\Local\Programs\Python\Python310\lib\site-packages\skl
earn\base.py:493: FutureWarning: The feature names should match those that w
ere passed during fit. Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.

  warnings.warn(message, FutureWarning)
```