Credit Card Lead Prediction

Happy Customer Bank is a mid-sized private bank that deals in all kinds of banking products, like Savings accounts, Current accounts, investment products, credit products, among other offerings.

The bank also cross-sells products to its existing customers and to do so they use different kinds of communication like tele-calling, e-mails, recommendations on net banking, mobile banking, etc.

In this case, the Happy Customer Bank wants to cross sell its credit cards to its existing customers. The bank has identified a set of customers that are eligible for taking these credit cards.

Now, the bank is looking for your help in identifying customers that could show higher intent towards a recommended credit card, given:

- Customer details (gender, age, region etc.)
- Details of his/her relationship with the bank (Channel_Code,Vintage, 'Avg_Asset_Value etc.)

Data Dictionary

Train Data

Variable	Definition
ID	Unique Identifier for a row
Gender	Gender of the Customer
Age	Age of the Customer (in Years)
Region_Code	Code of the Region for the customers
Occupation	Occupation Type for the customer
Channel_Code	Acquisition Channel Code for the Customer (Encoded)

Vintage	Vintage for the Customer (In Months)
Credit_Product	If the Customer has any active credit product (Home loan, Personal loan, Credit Card etc.)
Avg_Account_Balance	Average Account Balance for the Customer in last 12 Months
Is_Active	If the Customer is Active in last 3 Months
Is_Lead(Target)	If the Customer is interested for the Credit Card 0 : Customer is not interested 1 : Customer is interested

Test Data

Variable	Definition
ID	Unique Identifier for a row
Gender	Gender of the Customer
Age	Age of the Customer (in Years)
Region_Code	Code of the Region for the customers
Occupation	Occupation Type for the customer
Channel_Code	Acquisition Channel Code for the Customer (Encoded)
Vintage	Vintage for the Customer (In Months)
Credit_Product	If the Customer has any active credit product (Home loan, Personal loan, Credit Card etc.)
Avg_Account_Balance	Average Account Balance for the Customer in last 12 Months
Is_Active	If the Customer is Active in last 3 Months

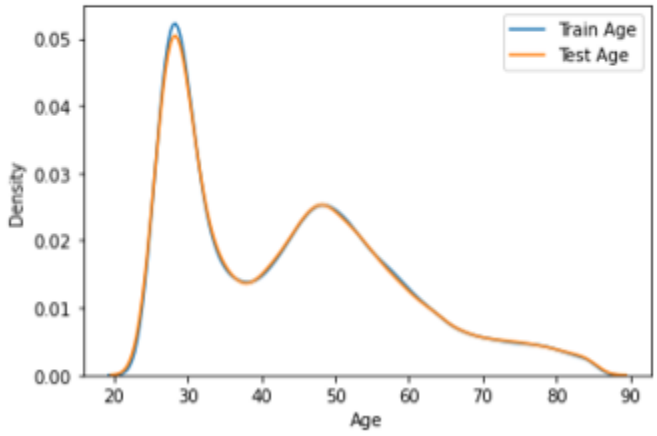
- We have train set of shape 245721 x 11 & test set of shape 105312 x 10

Data Discrepancies

- Variables in both the set has right data types
- There are approximately 11% missing values in Credit Product variable of both the sets
- **Data Quality Check : We need to check the distribution of both training set and test set be similar so that model performs well during production and there is no cause for data drift.**
    - Gender variable has same distribution for both the sets
    - Occupation variable of both the datasets proportion are similar
    - Proportions of Channel Code Variable in both the datasets are similar
    - Proportions of Credit Product Variable in both the datasets are similar
    - Proportion of Is Active Variable in both the datasets are similar
    - Distribution of variable Age in both the dataset is overlapping that is very similar, we see there are presence of at least three gaussians, i.e data consist of at least 3 clusters. Also there also lies positive skewness[1]

```
sns.distplot(train['Age'],hist=False,label='Train Age')
sns.distplot(test['Age'],hist=False,label='Test Age')
plt.legend()
```

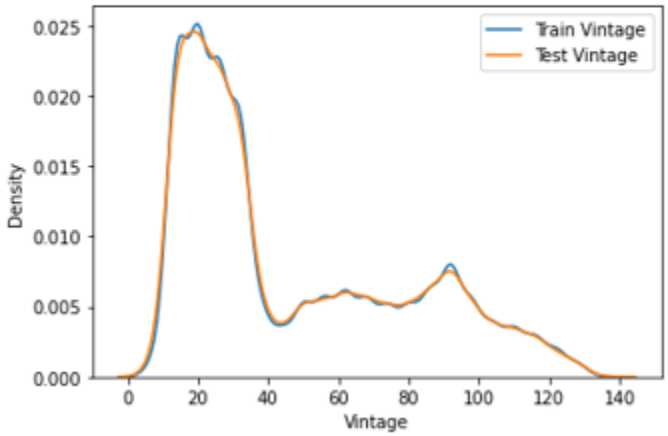
```
<matplotlib.legend.Legend at 0x18e9e0ba6d0>
```



- Distribution of the Vintage variable of the test set almost approximates the distribution of Vintage of the training set. We also see presence of at least 3 Gaussians, which indicate dataset consist of data from 3 different clusters. Also, we see there lies positive skewness[2]

```
sns.distplot(train['Vintage'],hist=False,label='Train Vintage')
sns.distplot(test['Vintage'],hist=False,label='Test Vintage')
plt.legend()
```

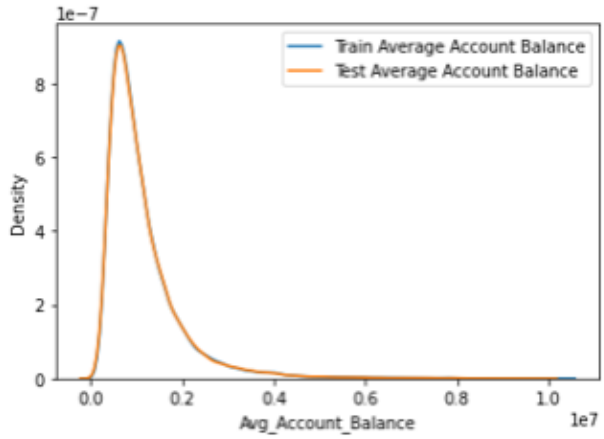
```
<matplotlib.legend.Legend at 0x18e9e0ef310>
```



- Distribution of Average Account Balance variable of test set almost approximate distribution of Average Account Balance variable of train set. We also see there lies positive skewness[3]

```
sns.distplot(train['Avg_Account_Balance'],hist=False,label='Train Average Account Balance')
sns.distplot(test['Avg_Account_Balance'],hist=False,label='Test Average Account Balance')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x18e9e1b77c0>
```



- So far we have seen that test dataset is quite a good approximation of training set and while in production there won't be issue faced like **data drift**

- [1] Age column of both the dataset is positively skewed, skewness value is .61
- [2] Vintage column of both the dataset is positively skewed, skewness value is .79
- [3] Average Account Balance column of both the dataset is highly positively skewed, skewness value is 2.9
- We see presence of outliers in Average Account Balance column for both the dataset, outliers for the same lies in with respect to interest of the customers as well
- We notice outliers for vintage variable in customers who did not show their interest in banks product
- We notice outliers for Age variable in customers who have shown their interest in banks product

Anomaly Treatment

- Treated outliers via quantile based bounds
- Since the Credit Product being the categorical column, has imputed the missing values with mode
- Skewness of variable : Age is reduced to .03, but it has increased the number of gaussians to at least 4
- Skewness of variable : Vintage is reduced to .10, but it increases the number of gaussians , at least presence of 5 gaussians are there
- Skewness of variable : Average Account Balance is reduced to .4, but it increases presence of gaussians to at least 3

Inference on Distribution of Target Column : The target column is quite imbalanced of ratio of 75:25 of Not Interested to Interested. Using SMOTE has increased the ratio to 65:35, which is still acceptable.

Data Insights :

- Out of 54% Male, only 14% had shown their interest for a recommended credit card, Whereas, out of 46% Females, only 9% had shown their interest. Which means a small chunk of the sample of the population shows their interest. It is quite realistic as generally we bounce of such promotional calls and also mark it spam

```
pd.crosstab(train['Gender'],train['Is_Lead'],normalize=True).sort_values(by=[1],ascending=False)
```

Is_Lead	0	1
Gender		
Male	0.401317	0.144802
Female	0.361479	0.092402

- 'RG268', 'RG283', 'RG284', 'RG254', 'RG280' are the top 5 region codes which have the highest percentage of interested people.[4]

```
pd.crosstab(train['Region_Code'],train['Is_Lead'],normalize=True).sort_values(by=[1],ascending=False).head()
```

Is_Lead	0	1
Region_Code		
RG268	0.102393	0.043838
RG283	0.083554	0.036159
RG284	0.054957	0.023869
RG254	0.085980	0.023250
RG280	0.039744	0.012246

- It has been seen that mostly self-employed people show higher interest than others.
    - Out of 40% of self employed only 11% showed their interest for a recommended credit card.
    - Out of 27% of people of Others occupation type, only 6% showed their interest for a recommended credit card

- Out of 29% salaried class people, only 4% showed their interest for a recommended credit card

```
pd.crosstab(train['Occupation'],train['Is_Lead'],normalize=True).sort_values(by=[1],ascending=False
```

Is_Lead	0	1
Occupation		
Self_Employed	0.297264	0.113299
Other	0.215627	0.069953
Salaried	0.246222	0.046781
Entrepreneur	0.003683	0.007171

- It has been seen as an acquisition channel partner for customers, out of 27% of channel partner ID : X3 , only 10% conversion was there , i.e only 10% customers showed interest out of 27% customers of total which were contacted by X3 channel partner. Channel Partner X2 only managed to get conversion of 9% out of 27% , Whereas, Channel partner X1 contacted maximum customers i.e 41% of total but their conversion was very low, i.e only 3%. Channel Partner X4 contacted the lowest % of customers of total , i.e only 1% and only managed to convert .5%.

```
pd.crosstab(train['Channel_Code'],train['Is_Lead'],normalize=True).sort_values(by=[1],ascending=False)
```

Is_Lead	0	1
Channel_Code		
X3	0.177002	0.102633
X2	0.185243	0.090371
X1	0.383504	0.038584
X4	0.017048	0.005616

- It has been seen that people who do not have any credit product are more interested in the recommended credit card than those who have an existing credit product. Out of 71% of people who did not have credit product only 14% showed the interest, where as out of 29% if people who had credit product only 9% showed their interest

```
pd.crosstab(train['Credit_Product'],train['Is_Lead'],normalize=True).sort_values(by=[1],ascending=Fals
```

Is_Lead	0	1
Credit_Product		
No	0.561950	0.144868
Yes	0.200846	0.092336

- It has been seen that customers who have not been active in the last 3 months with the bank , showed more interest than those who were active. Out of 61% inactive customers,12% showed their interest. Whereas, out of 39% active customers,10% customers showed their interest.

```
pd.crosstab(train['Is_Active'],train['Is_Lead'],normalize=True).sort_values(by=[1],ascending=False)
```

Is_Lead	0	1
Is_Active		
No	0.484314	0.127307
Yes	0.278483	0.109897

- We see those who were interested had higher average balance as compared to those who were not interested. Active customers have a higher average balance than non active customers.

```
pd.crosstab(train['Is_Active'],train['Is_Lead'],train['Avg_Account_Balance'],aggfunc='mean',normalize=
```

Is_Lead	0	1
Is_Active		
Yes	0.251032	0.259756
No	0.237890	0.251322

- We see those who expressed their interest in credit card , their average age is more than those who were not interested.Also, active customers average age is more than non active customers

```
pd.crosstab(train['Is_Active'],train['Is_Lead'],train['Age'],aggfunc='mean',normalize=True).sort_value
```

Is_Lead	0	1
Is_Active		
Yes	0.249552	0.254787
No	0.242330	0.253331

- We see that customers who have expressed their interest for the credit card have higher average vintage as compared to those who were not interested. Also, active customers are seen to be having higher average vintage than non active.

```
Active'],train['Is_Lead'],train['Vintage'],aggfunc='mean',normalize=True).sort_values(by=[1],ascending=
```

Is_Lead	0	1
Is_Active		
Yes	0.245260	0.269703
No	0.227685	0.257353

- We have seen that customers who expressed their interest have a higher average balance . It is also interesting to notice that customers who have existing credit products have a high average account balance.

```
pd.crosstab(train['Credit_Product'],train['Is_Lead'],train['Avg_Account_Balance'],aggfunc='mean',norma
```

Is_Lead	0	1
Credit_Product		
Yes	0.249854	0.256999
No	0.239650	0.253497

- We have already established that customers who expressed interest have higher average age than those who did not show interest. It is also interesting that those who have existing credit products, their average age is higher than those who did not have any existing credit products

```
pd.crosstab(train['Credit_Product'],train['Is_Lead'],train['Age'],aggfunc='mean',normalize=True).sort_
```

Is_Lead	0	1
Credit_Product		
Yes	0.248298	0.254037
No	0.243733	0.253931

- In addition to above, it is also interesting to know those who did not have existing credit product are having high average vintage value as compared to those who have credit product

```
pd.crosstab(train['Credit_Product'],train['Is_Lead'],train['Vintage'],aggfunc='mean',normalize=True).s
```

Is_Lead	0	1
Credit_Product		
No	0.232747	0.263786
Yes	0.239945	0.263522

- We already have established a baseline that those who have shown interest have higher average account balance. What is interesting to know is that the average balance of customers contacted by channel partner X3 are higher than others. That means, it won't be wrong to say that those customers have higher market spent share than others.

```
pd.crosstab(train['Channel_Code'],train['Is_Lead'],train['Avg_Account_Balance'],aggfunc='mean',normali
```

Is_Lead	0	1
Channel_Code		
X3	0.131649	0.135474
X2	0.126104	0.127505
X1	0.119521	0.125129
X4	0.118124	0.116493

- In addition to above, it is also interesting to know that X3 & X2 channel partners target customers of higher average age.

```
pd.crosstab(train['Channel_Code'],train['Is_Lead'],train['Age'],aggfunc='mean',normalize=True).sort_va
```

Is_Lead	0	1
Channel_Code		
X3	0.128388	0.128550
X2	0.126881	0.127012
X4	0.124385	0.124758
X1	0.117379	0.122647

- Also, in addition to above , X3 and X2 targets customers higher average vintage customers , where as X4 target least average vintage customers

```
pd.crosstab(train['Channel_Code'],train['Is_Lead'],train['Vintage'],aggfunc='mean',normalize=True).sort
```

Is_Lead	0	1
Channel_Code		
X3	0.149835	0.154642
X2	0.137596	0.142445
X1	0.113471	0.121013
X4	0.090554	0.090443

- We already have established a baseline that those who have shown interest have higher average account balance. What is interesting to know is that entrepreneur have higher average balance and salaried have least

```
pd.crosstab(train['Occupation'],train['Is_Lead'],train['Avg_Account_Balance'],aggfunc='mean',normalize=
```

Is_Lead	0	1
Occupation		
Entrepreneur	0.134100	0.134872
Other	0.122640	0.128786
Self_Employed	0.119961	0.123313
Salaried	0.114145	0.122184

- Also to add to above, Salaried class customers and Entrepreneurs are younger as compared to others and self employed.

```
pd.crosstab(train['Occupation'],train['Is_Lead'],train['Age'],aggfunc='mean',normalize=True).sort_valu
```

Is_Lead	0	1
Occupation		
Other	0.125786	0.130174
Self_Employed	0.125156	0.126567
Entrepreneur	0.126338	0.126483
Salaried	0.116230	0.123265

- To add further, entrepreneurs are of high vintage group and salaried of least.

```
pd.crosstab(train['Occupation'],train['Is_Lead'],train['Vintage'],aggfunc='mean',normalize=True).sort_
```

Is_Lead	0	1
Occupation		
Entrepreneur	0.128120	0.138813
Other	0.121453	0.136191
Self_Employed	0.123933	0.131713
Salaried	0.101909	0.117866

- 'RG284', 'RG283', 'RG268', 'RG254', 'RG253' are the region codes which are the top 5 regions having maximum average balance. It states there is high probability of conversion from these regions, which is being confirmed in previous point [4]

```
pd.crosstab(train['Region_Code'],train['Is_Lead'],train['Avg_Account_Balance'],aggfunc='mean',normaliz
```

Is_Lead	0	1
Region_Code		
RG284	0.017453	0.017891
RG283	0.017514	0.017820
RG268	0.017438	0.017811
RG254	0.017217	0.017601
RG253	0.017003	0.017072

- 'RG283', 'RG268', 'RG284', 'RG254', 'RG276' these are the region codes having higher average groups

```
pd.crosstab(train['Region_Code'],train['Is_Lead'],train['Age'],aggfunc='mean',normalize=True).sort_valu
```

Is_Lead	0	1
Region_Code		
RG283	0.014452	0.014740
RG268	0.014453	0.014739
RG284	0.014448	0.014734
RG254	0.014117	0.014684
RG276	0.014264	0.014678

- 'RG284', 'RG283', 'RG268', 'RG254', 'RG282' are the region codes having high vintage customers

```
pd.crosstab(train['Region_Code'],train['Is_Lead'],train['Vintage'],aggfunc='mean',normalize=True).sort_
```

Is_Lead	0	1
Region_Code		
RG284	0.014906	0.016357
RG283	0.014910	0.016343
RG268	0.014941	0.016335
RG254	0.013978	0.015729
RG282	0.013548	0.015576

- We already have established a baseline that those who have shown interest have higher average account balance. It is evident females are more conservative and they save more than more, hence their average balance is more

```
pd.crosstab(train['Gender'],train['Is_Lead'],train['Avg_Account_Balance'],aggfunc='mean',normalize=Tru
```

Is_Lead	0	1
Gender		
Female	0.241068	0.256863
Male	0.246106	0.255963

- In addition to above, females are comparatively younger than males

```
pd.crosstab(train['Gender'],train['Is_Lead'],train['Age'],aggfunc='mean',normalize=True).sort_values(by
```

Is_Lead	0	1
Gender		
Male	0.247748	0.255465
Female	0.243290	0.253497

- To add further, males are of more vintage than females.

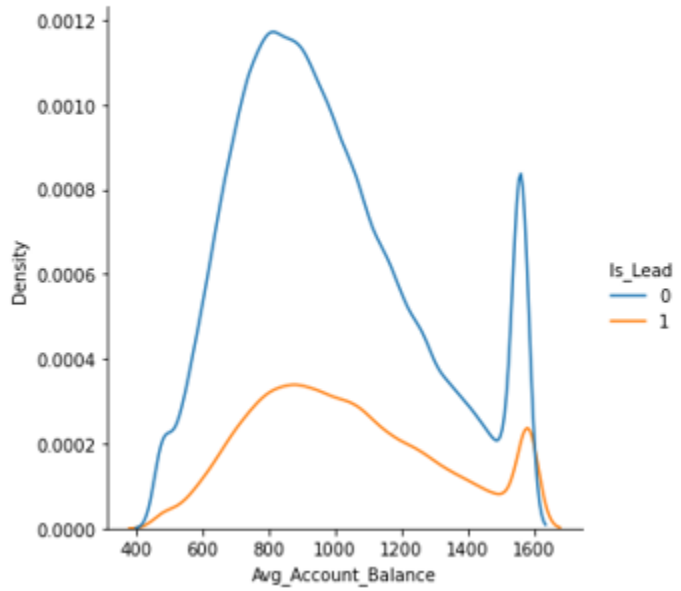
```
pd.crosstab(train['Gender'],train['Is_Lead'],train['Vintage'],aggfunc='mean',normalize=True).sort_value
```

Is_Lead	0	1
Gender		
Male	0.241887	0.269141
Female	0.229659	0.259313

- Distribution of balance with respect to the target variable indicates that variance is less hence the distribution is narrow. Height of the kde curve of interested people is small because of class imbalance

```
sns.displot(data=train,x='Avg_Account_Balance',hue='Is_Lead',kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x21196ef1760>
```

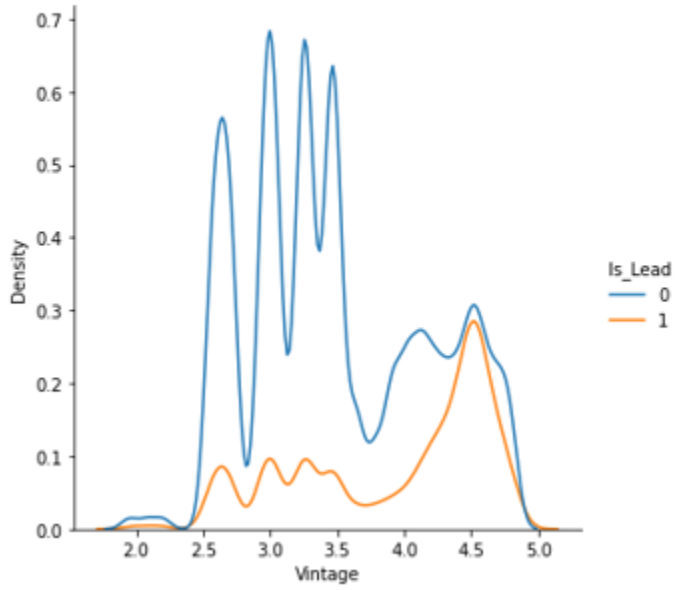


- We see in the distribution of vintage with respect to target variable , there is almost same distribution for range of vintage between 4.3 to 5. In the rest due to

lack of examples of class ==1 , there lies huge gap between the distribution

```
sns.displot(data=train,x='Vintage',hue='Is_Lead',kind='kde')
```

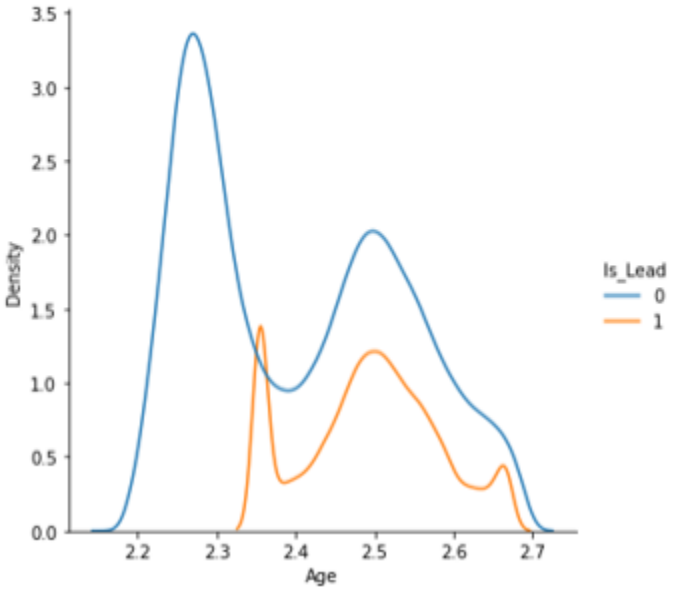
```
<seaborn.axisgrid.FacetGrid at 0x21194a72d30>
```

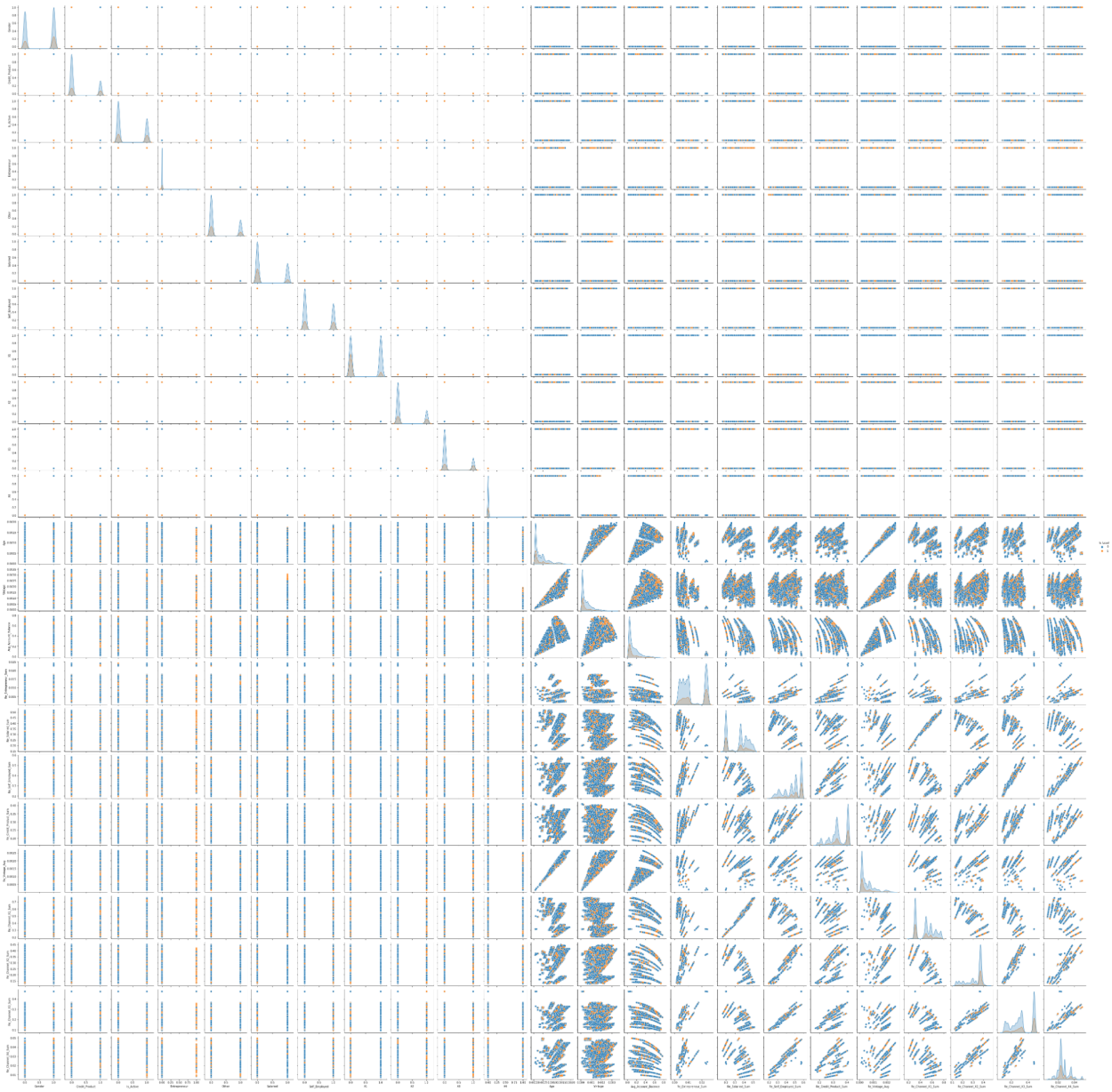


- There is a difference in the range of age for the customers who were not interested and for those who were interested. Variance in the distribution is less.

```
sns.displot(data=train,x='Age',hue='Is_Lead',kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x211954d6c40>
```





In the above graph we see univariate and bivariate distributions of variables. The data can not be separated by linear model classification models decision boundary. Hence we need to use algorithms like Decision Tree, Bagging and Boosting Models, if required neural network. Approach will try to achieve best results from an easy to interpret and less computational expensive model. For that, we shall set a baseline model.

Encoding Correction:

- Have used one-hot-encoding since the labels were not ordinal , also for few variables like gender just have done label encoding

Splitting Data: 80:20

Model Building :

Name	Train AUC	Validation AUC	Overfitting	Underfitting	Train Fold AUC	Val Fold AUC
Baseline-Decision Tree	.997	.67	True	-	-	-
Decision Tree-Tuned	.84	.83	-	-	-	-
Decision Tree - CV	.843	.834	-	-	.843	.843
XGBoost - Baseline	.88	.866	True			
XGBOOST-Tuned	.87	.86	-	-	-	-
XGBOOST - CV	.9121	.91	-	-	.91	.91
LGBM-Baseline	.868	.861	-	-	-	-
LGBM-Tuned	.878	.864	-	-	-	-
LGBM-CV	.92	.912	-	-	.913	.91

- We have got similar results in XGBOOST-CV and LIGHTGBM-CV. Out of the two i have taken LGBM because of less time taken by it. We can blend the two models but, it will make computationally expensive comparatively. Also to

highlight: number of iteration for XGBOOST-CV was 2714 but for LGBM-CV was 4209. Since iteration of light gbm was on higher side, still it was fast,

Visit the below given link for the solution :

- https://github.com/RiseAboveAll/JobaThon-Credit-Card