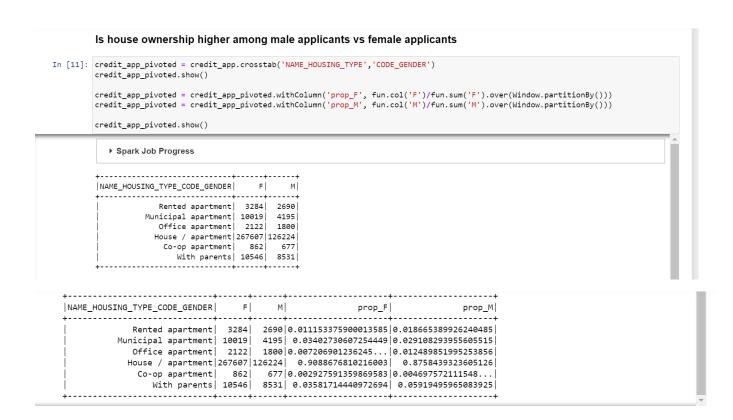
Credit Card Application Approval Model Assignment EDA

Questions & Answers

What is the proportion of females in the applicant customer base?

Based on above result 67.13 % is the ans.



We observe that 267607 females owns house as compared to male its 126224. The data shows house ownership is higher in Females.

Is there any correlation between income levels and education level?

Data shows positive corelation between income levels and education level

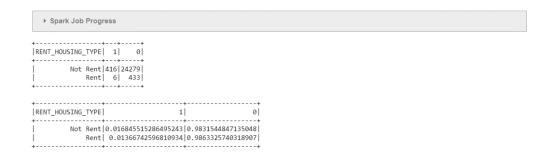
What is the average and median salary of the applicant base?

Average mean salary of applicant base is 160780.5

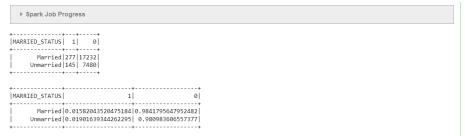
Do people owning cars have higher probability of bad customers?

We observe that people owning card has higher number of defaulters although the proportion wise when we compare to entire space the difference is marginal.

Do people living on rent have a higher proportion of bad customers as compared to the rest of the population



Considering the above data people living on rate are less likely to default as compared to non-rental person.



As per above data Married people are more likely to default as compared to unmarried.

Training Dataset Count: 17541 Test Dataset Count: 7593

targe	t prediction	probability
10	10.0	[0.9846176278381278,0.015382372161872151]
0	0.0	[0.9846176278381278,0.015382372161872151]
0	0.0	[0.986773310087667,0.01322668991233313]
0	0.0	[0.9863673224127191,0.013632677587280992]
0	0.0	[0.9863673224127191,0.013632677587280992]
0	0.0	[0.9872500834417228,0.012749916558277178]
0	0.0	[0.9872500834417228,0.012749916558277178]
0	0.0	[0.990614606918368,0.00938539308163209]
0	0.0	[0.990614606918368,0.00938539308163209]
0	0.0	[0.990614606918368,0.00938539308163209]
0	0.0	[0.990614606918368,0.00938539308163209]
0	0.0	[0.990614606918368,0.00938539308163209]
0	0.0	[0.990614606918368,0.00938539308163209]
0	0.0	[0.9849549125665301,0.015045087433469898]
0	0.0	[0.9903662030769003,0.009633796923099743]
0	0.0	[0.9903662030769003,0.009633796923099743]
0	0.0	[0.9903662030769003,0.009633796923099743]
0	0.0	[0.9903662030769003,0.009633796923099743]
0	0.0	[0.9903662030769003,0.009633796923099743]
0	0.0	[0.9855085313022302,0.014491468697769775]
+	-+	.+

only showing top 20 rows

targ	get predict	tion probability prob_int
0	0.0	[0.9846176278381278,0.015382372161872151] 0.015382372
0	0.0	[0.9846176278381278,0.015382372161872151] 0.015382372
0	0.0	[0.986773310087667,0.01322668991233313] 0.01322669
0	0.0	[0.9863673224127191,0.013632677587280992] 0.013632677
0	0.0	[0.9863673224127191,0.013632677587280992] 0.013632677
0	0.0	[0.9872500834417228,0.012749916558277178] 0.012749917
0	0.0	[0.9872500834417228,0.012749916558277178] 0.012749917
0	0.0	[0.990614606918368,0.00938539308163209] 0.009385393
0	0.0	[0.990614606918368,0.00938539308163209] 0.009385393
0	0.0	[0.990614606918368,0.00938539308163209] 0.009385393
0	0.0	[0.990614606918368,0.00938539308163209] 0.009385393
0	0.0	[0.990614606918368,0.00938539308163209] [0.009385393
0	0.0	[0.990614606918368,0.00938539308163209] [0.009385393
0	0.0	[0.9849549125665301,0.015045087433469898] 0.015045088
0	0.0	[0.9903662030769003,0.009633796923099743] 0.009633797
0	0.0	[0.9903662030769003,0.009633796923099743] 0.009633797
0	0.0	[0.9903662030769003,0.009633796923099743] 0.009633797
0	0.0	[0.9903662030769003,0.009633796923099743] 0.009633797
0	0.0	[0.9903662030769003,0.009633796923099743] 0.009633797
0	0.0	[0.9855085313022302,0.014491468697769775] 0.014491469

only showing top 20 rows

```
In [28]: # AUC ROC for training set
trainingSummary = model.summary
    roc = trainingSummary = model.summary
    roc = trainingSummary = model.summary
    roc = trainingSummary = model.summary
    rot = trainingSummary = model.summary
    rot = trainingSummary = model.summary
    plt.splot(prive Positive Rate')
    plt.slabel('True Positive Rate')
    plt.slabel('Talse Positive Rate')
    plt.show()
    print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))

# AUC ROC for test set
    preds = predict_test.select('target','prob_int').rdd.map(lambda row: (float(row['prob_int']), float(row['target']))).collect()
    ln_probs, y_test = zip('preds)
    fpr, tpr, thresholds = roc_curve(y_test, ln_probs, pos_label = 1)

    ns_probs = [0 for _ in range(len(y_test))]
    # calculate scores
    ns_auc = roc_auc_score(y_test, ns_probs)
    ln_auc = roc_auc_score(y_test, ln_probs)

# summarize scores
    print('Testing set areaUnderROC=%.3f' % (ln_auc))
    # calculate roc curves
    ns_fpr, ns_tpr, _ = roc_curve(y_test, ln_probs)

    ln_fpr, in_tpr, _ = roc_curve(y_test, ln_probs)

    ln_fpr, in_tpr, _ = roc_curve(y_test, ln_probs)

# plot the roc curve for the model

    pyplot.plot(ns_fpr, ns_tpr, linestyles'--', label='No Skill')
    pyplot.plot(ns_fpr, ln_tpr, marker=', label='logistic')

# avis lobels

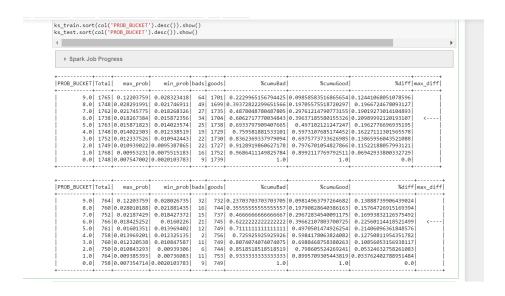
pyplot.ylabel('True Positive Rate')

# show the (egend
    pyplot.legend()
```

show the plot pyplot.show()

Spark Job Progress

Training set areaUnderROC: 0.6454687677330996 Testing set areaUnderROC=0.621



Model Evaluation 2nd Iteration

```
# AUC ROC for test set
preds = predict_test.select('target','prob_int').rdd.map(lambda row: (float(row['prob_int']), float(row['target']))).collect()
lr.probs, y.test = zip("preds)
fpr, tpr, thresholds = roc_curve(y_test, lr_probs, pos_label = 1)
ns_probs = [0 for _in range(len(y_test))]
# calculate scores
ns_auc = roc_auc_score(y_test, ns_probs)
lr_auc = roc_auc_score(y_test, lr_probs)
# summarize scores
print('Testing set arealunderROC*&3f' % (lr_auc))
# calculate roc curves
ns_fpr, ns_tpr, _= roc_curve(y_test, lr_probs)
lr_fpr, lr_tpr, _= roc_curve(y_test, lr_probs)
# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.ylabel('True Positive Rate')
# axis labels
pyplot.xlabel('True Positive Rate')
# show the lagend()
# show the plot
pyplot.show()
```

Training Dataset Count: 17541
Test Dataset Count: 7593
Training set areaUnderROC: 0.6521478229155769
Testing set areaUnderROC=0.649

+				+	+				+
max_diff	%diff	%cumuGood	%cumuBad	goods	bads	min_prob	max_prob	Total	PROB_BUCKET
 	0.12081044480318456	0.09870175031876666	0.21951219512195122	1703	+ 63	0.027494008	0.9828491	1766	9.0
İ	0.1829670158795678	0.19682392488698272	0.3797909407665505	1693	46	0.021537388	0.027488615	1739	8.0
	0.2114463989363271						0.021510385		
	0.24195429712001337 0.23570638975196984						0.017968645 0.015847946		
	0.18004046125344253						0.013699586		
	0.1417961355423718		0.8397212543554007				0.012264656		
	0.09096128393597769		0.8954703832752613				0.010988842		
:		0.8985742436536456					0.009227373		
1	0.0	1.0	1.0	1750	13	0.001/910189	0.0076742047	1/63	0.0

max_diff	%diff	%cumuGood	%cumuBad	goods	bads	min_prob	max_prob	Total	PROB_BUCKET
	0.15370221387920502 0.18821548821548822	0.19696969696969696	0.3851851851851852	737	18	0.021611314	0.99120146 0.027115295	755	9.0
<	0.22192425732248738 0.2708500938589434 0.22200470784541587	0.39581657280772325	0.66666666666666	740	20	0.015905177	0.021602597 0.018163167 0.015900783	760	7.0 6.0 5.0
	0.1658859986293615 0.11690652841095317	0.5970769643336015 0.6979082864038616	0.762962962962963 0.8148148148148148	750 752	6 7	0.012186022 0.0109289065	0.013618922 0.012183561	756 759	4.0
	0.07613897082923637 0.0410357259914782 0.0		0.8740740740740741 0.9407407407407408 1.0		9	0.0073632	0.010906844 0.009193477 0.007326404	768	1.0

```
In [32]: #Creating prediction class based on the prob cut off by KS value

min_prob_max_ks = ks_test.where(col("max_diff") == '<---').select(col("min_prob")).collect()[0]

predict_test = predict_test.withColumn('pred_class', F.when(F.col('prob_int') > round(min_prob_max_ks[0],4) ,1).otherwise(0))

predict_test.groupBy('pred_class', 'target').count().show()

#Generating evaluation metrics

results = predict_test.select[['pred_class', 'target']])

results = results.withColumn("target", results("target"].cast(DoubleType())))

results = results.withColumn("pred_class", results("pred_class"].cast(DoubleType())))

predictionAndlabels=results.rdd

metrics = MulticlassStetrics(predictionAndlabels)

metrics1 = BinaryClassificationMetrics(predictionAndlabels)

cm=metrics.confusionMatrix().toArray()

print(cm)

# Overall statistics

precision = metrics.precision(1)

recall = metrics.precision(1)

recall = metrics.precision = %s" % precision)

print("Precision = %s" % metrics.areaUnderPR)

# Area under Precision-recall curve

print("Area under PR = %s" % metrics1.areaUnderPR)

# Area under ROC curve

print("Area under ROC = %s" % metrics1.areaUnderPROC)
```