Models/Results

Standard Scalar, Train-Test Split Logistic Regression on Numerical Columns

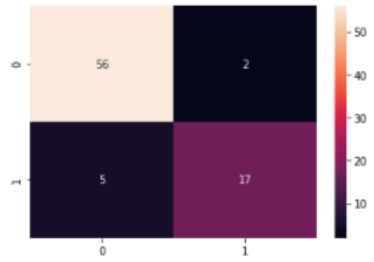
```
y_pred_test = lr.predict(X_test)

cm = confusion_matrix(y_test,y_pred_test)
sns.heatmap(cm, annot = True, fmt = 'd')
print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94        58
           1       0.89      0.77      0.83        22

    accuracy                           0.91        80
   macro avg       0.91      0.87      0.89        80
weighted avg       0.91      0.91      0.91        80
```



	Gender	Age	EstimatedSalary	Purchased
395	0	46	41000	1
396	1	51	23000	1
397	0	50	20000	1
398	1	36	33000	0
399	0	49	36000	1

Standard Scalar, Train-Test Split Logistic Regression on WoE Encoded Columns

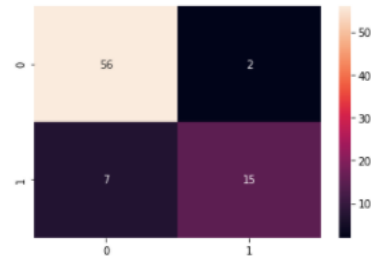
```
y_pred_test = lr.predict(X_test)

cm = confusion_matrix(y_test,y_pred_test)
sns.heatmap(cm, annot = True, fmt = 'd')
print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.89      0.97      0.93        58
           1       0.88      0.68      0.77        22

    accuracy                           0.89        80
   macro avg       0.89      0.82      0.85        80
weighted avg       0.89      0.89      0.88        80
```



	Purchased	age_WoE_ENC	sal_WoE_ENC	gender_WoE_ENC
380	0	0.010783	0.000799	0.912426
381	1	0.233636	0.017191	0.912426
382	1	0.017972	0.377413	1.089643
383	1	0.071888	0.017191	0.912426
384	1	0.000000	0.017191	1.089643
385	1	0.000000	0.000799	0.912426
386	1	0.071888	0.017191	1.089643

DataFrame mapping technique used in 2nd model:

Note: WoE labels for each age (not grouped)

	Total Count	Purchased	Not Purchased	%P	%NP	WOE	IV
41	16	1.0	15.0	0.70%	5.84%	0.001198	-0.006155
42	16	6.0	10.0	4.20%	3.89%	0.010783	0.003286
43	3	2.0	1.0	1.40%	0.39%	0.035944	0.036285
44	2	1.0	1.0	0.70%	0.39%	0.017972	0.005575
45	7	6.0	1.0	4.20%	0.39%	0.107832	0.410485
46	12	7.0	5.0	4.90%	1.95%	0.025161	0.074214
47	14	12.0	2.0	8.39%	0.78%	0.107832	0.820969
48	14	13.0	1.0	9.09%	0.39%	0.233636	2.033058
49	10	8.0	2.0	5.59%	0.78%	0.071888	0.346227
50	4	3.0	1.0	2.10%	0.39%	0.053916	0.092132

Target Salary Groupings used for WoE tier-marked labels (grouped):
["15k-26k","26k-40k","40k-55k","55k-67k","67k-81k","81k-100k", "100k-120k","120k-135k","135k-150k"]

	Total Count	Purchased	Not Purchased	%P	%NP	WOE	IV
100k-120k	32	24	8	16.78%	3.11%	0.053916	0.737053
120k-135k	19	17	2	11.89%	0.78%	0.152762	1.697173
135k-150k	22	21	1	14.69%	0.39%	0.377413	5.395569
15k-26k	42	13	29	9.09%	11.28%	0.008056	-0.017669
26k-40k	45	22	23	15.38%	8.95%	0.017191	0.110625
40k-55k	61	10	51	6.99%	19.84%	0.003524	-0.045287
55k-67k	47	2	45	1.40%	17.51%	0.000799	-0.012869
67k-81k	79	13	66	9.09%	25.68%	0.003540	-0.058728
81k-100k	53	21	32	14.69%	12.45%	0.011794	0.026348

Final map before DataFrame reduction:

	Gender	Age	EstimatedSalary	Purchased	Salary_Group	age_WoE_ENC	sal_WoE_ENC	gender_WoE_ENC
380	1	42	64000	0	55k-67k	0.010783	0.000799	0.912426
381	1	48	33000	1	26k-40k	0.233636	0.017191	0.912426
382	0	44	139000	1	135k-150k	0.017972	0.377413	1.089643
383	1	49	28000	1	26k-40k	0.071888	0.017191	0.912426
384	0	57	33000	1	26k-40k	0.000000	0.017191	1.089643
385	1	56	60000	1	55k-67k	0.000000	0.000799	0.912426
386	0	49	39000	1	26k-40k	0.071888	0.017191	1.089643
387	1	39	71000	0	67k-81k	0.011981	0.003540	0.912426
388	1	47	34000	1	26k-40k	0.107832	0.017191	0.912426
389	0	48	35000	1	26k-40k	0.233636	0.017191	1.089643
390	1	48	33000	1	26k-40k	0.233636	0.017191	0.912426
391	1	47	23000	1	15k-26k	0.107832	0.008056	0.912426
392	0	45	45000	1	40k-55k	0.107832	0.003524	1.089643
393	1	60	42000	1	40k-55k	0.000000	0.003524	0.912426
394	0	39	59000	0	55k-67k	0.011981	0.000799	1.089643
395	0	46	41000	1	40k-55k	0.025161	0.003524	1.089643
396	1	51	23000	1	15k-26k	0.035944	0.008056	0.912426
397	0	50	20000	1	15k-26k	0.053916	0.008056	1.089643
398	1	36	33000	0	26k-40k	0.012837	0.017191	0.912426
399	0	49	36000	1	26k-40k	0.071888	0.017191	1.089643

Logistic Regression on one-hot encoded variables, K-fold split

```
X = features.iloc[:,1:]
y = features.iloc[:,0]
```

```
kfold = KFold(n_splits=5, random_state=0, shuffle=True)
model = LogisticRegression(solver='liblinear')
results = cross_val_score(model, X, y, cv=kfold)
y_pred = cross_val_predict(model, X, y, cv=kfold)
conf_matrix = confusion_matrix(y, y_pred)
report = classification_report(y, y_pred)
# Output the accuracy. Calculate the mean and std across all folds.
print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
print(conf_matrix)
print(report)
```

```
Accuracy: 87.250% (6.490%)
[[240  17]
 [ 34 109]]
              precision    recall  f1-score   support

           0       0.88      0.93      0.90       257
           1       0.87      0.76      0.81       143

    accuracy                           0.87       400
   macro avg       0.87      0.85      0.86       400
weighted avg       0.87      0.87      0.87       400
```

Logistic Regression on one-hot encoded variables, Train-Test split

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=42)

model = LogisticRegression(solver='liblinear')
model.fit(X_train, y_train)
predicted = model.predict(X_test)
matrix = confusion_matrix(y_test, predicted)
report = classification_report(y_test, predicted)
print(matrix)
print(report)
```

```
[[49  3]
 [ 4 24]]
              precision    recall  f1-score   support

           0       0.92      0.94      0.93        52
           1       0.89      0.86      0.87        28

    accuracy                           0.91        80
   macro avg       0.91      0.90      0.90        80
weighted avg       0.91      0.91      0.91        80
```

Dataframe used for both models:

	Purchased	Gender_Female	Gender_Male	Age Group_18-25	Age Group_26-35	Age Group_36-45	Age Group_46-55	Age Group_55+	Salary Group_100k-120k	Salary Group_120k-135k	Salary Group_135k-150k	Salary Group_15k-26k	Gr
0	0	0	1	1	0	0	0	0	0	0	0	1	
1	0	0	1	0	1	0	0	0	0	0	0	1	
2	0	1	0	0	1	0	0	0	0	0	0	0	
3	0	1	0	0	1	0	0	0	0	0	0	0	
4	0	0	1	1	0	0	0	0	0	0	0	0	

Note: One-Hot Encoding for age (grouped):
["18-25","26-35","36-45","46-55","55+"]
Target Salary Groupings used for one-hot encoding (grouped):
["15k-26k","26k-40k","40k-55k","55k-67k","67k-81k","81k-100k", "100k-120k","120k-135k","135k-150k"]
Note: models not scaled, one-hot normalized distribution

Dataframe Mapping technique used for IV/WoE mapping in following models:

	Total Count	Purchased	Not Purchased	%P	%NP	WOE	IV
Female	204	77	127	53.85%	49.42%	1.089643	4.826911
Male	196	66	130	46.15%	50.58%	0.912426	-4.041875

	Total Count	Purchased	Not Purchased	%P	%NP	WOE	IV
18-25	49	0	49	0.00%	19.07%	0.000000	-0.000000
26-35	129	17	112	11.89%	43.58%	0.002728	-0.086452
36-45	119	38	81	26.57%	31.52%	0.008431	-0.041685
46-55	75	62	13	43.36%	5.06%	0.085713	3.282651
55+	28	26	2	18.18%	0.78%	0.233636	4.066116

	Total Count	Purchased	Not Purchased	%P	%NP	WOE	IV
100k-120k	32	24	8	16.78%	3.11%	0.053916	0.737053
120k-135k	19	17	2	11.89%	0.78%	0.152762	1.697173
135k-150k	22	21	1	14.69%	0.39%	0.377413	5.395569
15k-26k	42	13	29	9.09%	11.28%	0.008056	-0.017669
26k-40k	45	22	23	15.38%	8.95%	0.017191	0.110625
40k-55k	61	10	51	6.99%	19.84%	0.003524	-0.045287
55k-67k	47	2	45	1.40%	17.51%	0.000799	-0.012869
67k-81k	79	13	66	9.09%	25.68%	0.003540	-0.058728
81k-100k	53	21	32	14.69%	12.45%	0.011794	0.026348

Final Map before DataFrame reduction:

	Gender	Purchased	Age_Group	Salary_Group	age_WoE_ENC	sal_WoE_ENC	gender_WoE_ENC
380	Male	0	36-45	55k-67k	0.008431	0.000799	0.912426
381	Male	1	46-55	26k-40k	0.085713	0.017191	0.912426
382	Female	1	36-45	135k-150k	0.008431	0.377413	1.089643
383	Male	1	46-55	26k-40k	0.085713	0.017191	0.912426
384	Female	1	55+	26k-40k	0.233636	0.017191	1.089643
385	Male	1	55+	55k-67k	0.233636	0.000799	0.912426
386	Female	1	46-55	26k-40k	0.085713	0.017191	1.089643
387	Male	0	36-45	67k-81k	0.008431	0.003540	0.912426
388	Male	1	46-55	26k-40k	0.085713	0.017191	0.912426
389	Female	1	46-55	26k-40k	0.085713	0.017191	1.089643
390	Male	1	46-55	26k-40k	0.085713	0.017191	0.912426
391	Male	1	46-55	15k-26k	0.085713	0.008056	0.912426
392	Female	1	36-45	40k-55k	0.008431	0.003524	1.089643
393	Male	1	55+	40k-55k	0.233636	0.003524	0.912426
394	Female	0	36-45	55k-67k	0.008431	0.000799	1.089643
395	Female	1	46-55	40k-55k	0.085713	0.003524	1.089643
396	Male	1	46-55	15k-26k	0.085713	0.008056	0.912426
397	Female	1	46-55	15k-26k	0.085713	0.008056	1.089643
398	Male	0	36-45	26k-40k	0.008431	0.017191	0.912426
399	Female	1	46-55	26k-40k	0.085713	0.017191	1.089643

Results of WoE encoding log regression *without* standard scaling, K-fold split

Note: columns

```
kfold = StratifiedKFold(n_splits=5, random_state=0, shuffle=True)
model = LogisticRegression(solver='liblinear')
results = cross_val_score(model, X, y, cv=kfold)
y_pred = cross_val_predict(model, X, y, cv=kfold)
# Output the accuracy. Calculate the mean and std across all folds.
print("Accuracy, STD: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
print(results)
conf_matrix = confusion_matrix(y, y_pred)
report = classification_report(y, y_pred)
ks = cohen_kappa_score(y, y_pred)
log_loss_score = log_loss(y, y_pred)
print(conf_matrix)
print(report)
print(ks)
print(log_loss_score)
```

```
Accuracy, STD: 74.500% (3.758%)
[0.8    0.6875 0.725  0.75   0.7625]
[[253   4]
 [ 98  45]]
              precision    recall  f1-score   support

           0       0.72      0.98      0.83       257
           1       0.92      0.31      0.47       143

    accuracy                           0.74       400
   macro avg       0.82      0.65      0.65       400
weighted avg       0.79      0.74      0.70       400

0.35017360558086197
8.807395976676526
```

Results of WoE encoding log regression *with* standard scaling, K-fold split

```
X = woe_ad_clicks.iloc[:,1:]
y = woe_ad_clicks.iloc[:,0]
```

```
#note: using SStandard Scalar on WoE
ss = StandardScaler()
X = ss.fit_transform(X)
```

```
kfold = StratifiedKFold(n_splits=5, random_state=0, shuffle=True)
model = LogisticRegression(solver='liblinear')
results = cross_val_score(model, X, y, cv=kfold)
y_pred = cross_val_predict(model, X, y, cv=kfold)
# Output the accuracy. Calculate the mean and std across all folds.
print("Accuracy, STD: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
print(results)
conf_matrix = confusion_matrix(y, y_pred)
report = classification_report(y, y_pred)
ks = cohen_kappa_score(y, y_pred)
log_loss_score = log_loss(y, y_pred)
print(conf_matrix)
print(report)
print(ks)
print(log_loss_score)
```

```
Accuracy, STD: 87.750% (4.138%)
[0.8375 0.8375 0.875  0.95   0.8875]
[[241  16]
 [ 33 110]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91       257
           1       0.87      0.77      0.82       143

    accuracy                           0.88       400
   macro avg       0.88      0.85      0.86       400
weighted avg       0.88      0.88      0.88       400

0.7261192778491979
4.231032092273767
```

DataFrame used on both WoE models:

	Purchased	age_WoE_ENC	sal_WoE_ENC	gender_WoE_ENC
0	0	0.000000	0.008056	0.912426
1	0	0.002728	0.008056	0.912426
2	0	0.002728	0.003524	1.089643
3	0	0.002728	0.000799	1.089643
4	0	0.000000	0.003540	0.912426

Results of IV Encoding *without* standard scaling

In [109]:
```
kfold2 = KFold(n_splits=5, random_state=0, shuffle=True)
model2 = LogisticRegression(solver='liblinear')
results2 = cross_val_score(model, X2, y2, cv=kfold)
y_pred2 = cross_val_predict(model, X2, y2, cv=kfold)
# Output the accuracy. Calculate the mean and std across all folds.
print("Accuracy, STD: %.3f%% (%.3f%%)" % (results2.mean()*100.0, results2.std()*100.0))
print(results2)
conf_matrix2 = confusion_matrix(y2, y_pred2)
report2 = classification_report(y2, y_pred2)
log_loss_score2 = log_loss(y2, y_pred2)
print(conf_matrix2)
print(report2)
print(log_loss_score2)
```

```
Accuracy, STD: 87.750% (4.138%)
[0.8375 0.8375 0.875  0.95   0.8875]
[[241  16]
 [ 33 110]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91       257
           1       0.87      0.77      0.82       143

    accuracy                           0.88       400
   macro avg       0.88      0.85      0.86       400
weighted avg       0.88      0.88      0.88       400

4.231032092273767
```

Results of IV encoding *with* standard scaling

Log Regression using IV encoded categorical variables, k-fold split

In [220]:
```
X2 = iv_ad_clicks.iloc[:,1:]
y2 = iv_ad_clicks.iloc[:,0]
```

In [221]:
```
ss = StandardScaler()
X = ss.fit_transform(X2)
```

In [222]:
```
kfold2 = KFold(n_splits=5, random_state=0, shuffle=True)
model2 = LogisticRegression(solver='liblinear')
results2 = cross_val_score(model, X2, y2, cv=kfold)
y_pred2 = cross_val_predict(model, X2, y2, cv=kfold)
# Output the accuracy. Calculate the mean and std across all folds.
print("Accuracy, STD: %.3f%% (%.3f%%)" % (results2.mean()*100.0, results2.std()*100.0))
print(results2)
conf_matrix2 = confusion_matrix(y2, y_pred2)
report2 = classification_report(y2, y_pred2)
log_loss_score2 = log_loss(y2, y_pred2)
print(conf_matrix2)
print(report2)
print(log_loss_score2)
```

```
Accuracy, STD: 87.750% (4.138%)
[0.8375 0.8375 0.875  0.95   0.8875]
[[241  16]
 [ 33 110]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91       257
           1       0.87      0.77      0.82       143

    accuracy                           0.88       400
   macro avg       0.88      0.85      0.86       400
weighted avg       0.88      0.88      0.88       400

4.231032092273767
```

Reduced DataFrame used in both IV models:

	Purchased	age_IV_ENC	sal_IV_ENC	gender_IV_ENC
0	0	-0.000000	-0.017669	-4.041875
1	0	-0.086452	-0.017669	-4.041875
2	0	-0.086452	-0.045287	4.826911
3	0	-0.086452	-0.012869	4.826911
4	0	-0.000000	-0.058728	-4.041875