Document Project 1

1. **Data Description**

   The data is synthetic data made from the analysis of a few billion real U.S. applications over about 10 years. The goal is to find application/identity fraud, so only identity fields. The data covers the time of 2017. There are 10 fields and 1000000 records.

   - **Numerical Table**

Field name	% Populated	Min	Max	Mean	Stdev	% Zero
Date	100%	2017-01-01	2017-12-31	/	/	0.00

   - **Categorial Table**

Field name	% Populated	# Unique Values	Most Common Value
Record	100%	1000000	N/A
Firstname	100%	78136	EAMSTRMT
Lastname	100%	177001	ERJSAXA
Address	100%	829852	1775 XJXE LN
ssn	100%	852745	938972725
Zip5	100%	26370	68138
dob	100%	169240	19640318
homephone	100%	106755	6384782007
Fraud_label	100%	2	0

2. **Data Cleaning**

   This will just be transforming the frivolous fields, but describe why and how.

   a. Default Values

      We found default values the business put for replacing missing values to ensure they do not confuse our models. If we do not handle the default values, there might be same field values that appeared a lot, causing false positive results. There are 4 fields that contained default values, and we replaced them with unique record number to ensure they won't link to previous records in this field and influence our model. (1) SSN, we replace all the SSN that equal to '999999999'. (2) Address. Replace address with records equal '123 MAIN ST' (3) DOB. Replaced records with DOB equal '19070626' (4) Homephone. Replaced records with Homephone equal '9999999999'

   b. Time Value Columns

      Convert time columns (date and dob_dt) to pandas datetime format

   c. Data frame Format

      Make all the values of zip5 have five digit to achieve right alignment

3. **Variable Creation**

Description of Variables	# Variables Created
Original Fields from the dataset excluding 'record' and 'fraud_label'	8
Age when apply Applicant Age when application is submitted	1
Day of week target encoded day of week and average fraud percentage of that day	2
New Entities combining/concatenating different original fields	18
Velocity # records with the same entity over the last [0,1,3,7,14, 30] days	128
Relative Velocity The ratio of # records with entities from the last [0,1] days to the #records with that same group last [3,7,14,30] days	184
Day Since Variable # days since an application with that entity has been seen. Entities list is attached below	23
# unique entity for particular field over the last [0,1,3,7,14, 30, 60] days	3542
Maximum Indicator count of most common record for each entity in the last [1, 3, 7, 30] days	92
Age Indicator [max, mean, min] age for each entity	69

Entities: 'ssn', 'address', 'zip5','dob', 'homephone', 'name', 'fulladdress', 'name_dob','name_fulladdress', 'name_homephone','fulladdress_dob', 'fulladdress_homephone', 'dob_homephone', 'homephone_name_dob', 'ssn_firstname', 'ssn_lastname', 'ssn_address', 'ssn_zip5','ssn_dob','ssn_homephone', 'ssn_name', 'ssn_fulladdress','ssn_name_dob']

There are two main ways of identity fraud. (1) a fraudster has a list of victims' information and use this list applying for many products with many identities. He uses the victims' core identity information (SSN, Name, DOB) and his own contact information (Address, Phone number). (2) a victim's information was used by many fraudsters.

To catch fraudulent behaviors, we create variables on the table showed above. For the first mode of fraud, there will be many applications and identities (such as SSN) using the same contact information such as phone number and address. In this case, we can look at

variables such as # apps at that address (an example of Velocity) , #apps at that phone number, # SSNs at that phone, # addresses with the SSN to detect the fraud. For the second situation, there are many applications with the same identity information, but from different contact information. we can focus on variables like #Address at that SSN, #Phone at that SSN, # Address at that SSN_Name, and #app at that SSN. We can also look at days since an application with that entity has been seen to detect abnormal applications if an entity is using multiple times within a short time range.

wrapper order	variable	KS
1	zip5_count_1	0.221239028
2	fulladdress_count_0_by_30	0.290722131
3	ssn_firstname_day_since	0.226427511
4	fulladdress_day_since	0.333268536
5	address_unique_count_for_ssn_zip5_60	0.289723617
6	address_count_30	0.332648157
7	address_day_since	0.334139944
8	address_count_14	0.32243628
9	fulladdress_count_14	0.321952925
10	address_count_7	0.301735277
11	fulladdress_count_7	0.301666247
12	address_unique_count_for_name_homephone_60	0.292437957
13	address_count_0_by_30	0.291922189
14	address_unique_count_for_homephone_name_dob_60	0.291409788
15	fulladdress_unique_count_for_ssn_homephone_60	0.289990622
16	address_unique_count_for_ssn_name_60	0.289679212
17	fulladdress_unique_count_for_name_homephone_60	0.289535139
18	address_unique_count_for_ssn_homephone_60	0.289166405
19	fulladdress_unique_count_for_homephone_name_dob_60	0.288482719
20	fulladdress_unique_count_for_dob_homephone_60	0.288442887
21	address_unique_count_for_ssn_firstname_60	0.288127273
22	address_unique_count_for_ssn_name_dob_60	0.287644887
23	address_unique_count_for_dob_homephone_60	0.287555865
24	address_unique_count_for_ssn_lastname_60	0.287443597
25	address_unique_count_for_name_60	0.287411369
26	fulladdress_unique_count_for_ssn_name_60	0.286799367

27	fulladdress_unique_count_for_ssn_lastname_60	0.286776127
28	fulladdress_unique_count_for_ssn_60	0.286764374
29	fulladdress_unique_count_for_ssn_firstname_60	0.286763364
30	address_unique_count_for_ssn_60	0.285913355

4. **Feature selection**

   Feature selection is a crucial step in the process of building predictive models. It involves identifying and selecting the most important features or variables from a dataset that are likely to have the most significant impact on the outcome or response variable. It helps to resolve the problem of curse of dimensionality for nonlinear models and reduce the computational cost of modeling, speeding the training process.

   There are three main ways of feature selections: Filter, Wrapper, and Embedded.

   a. I first ran the filter, a univariate model performance measure, on all the variables. I perform the Kolmogorov-Smirnov (KS) test to see whether the distribution of a variable in a group of "good" samples is significantly different from the distribution of the same variable in a group of "bad" samples and used the returned statistics as the filter score. I sort the variables according to the filter score on descending order, and I kept top 300 variables out of 2242 variables after filtering.
   b. For the wrapper part, we used LGBM, which is fast and its hyperparameter is set to be simple, to conduct forward selections on the remaining part of variables. We then used FDR @ 3%, which is the fraud detection rate at the top 3% records, as performance metric for the model. The FDR Score @ 3% achieves 6% starting on the first five variables; and we kept first 30 variables as a way to compare the performance of different hyperparameters and different number of variables for later preliminary modeling analysis.
   c. Embedded method is within the algorithms. For instance, there are embedded feature selection in decision tree, and we can also use lasso regularization as the feature selection method.

5. **Preliminary Models Exploration**

Model	Parameters								Average FDR at 3%		
	Iteration	Penalty	C	Solver	l1 ratio	Num_of_variables			Train	Test	OOT
Logistics Regression	1	L2	1	lbfgs	none	5			0.478	0.474	0.463
	2	L1	1	liblinear	none	10			0.490	0.482	0.474
	3	L2	1	liblinear	none	10			0.488	0.487	0.476
	4	L2	3	lbfgs	none	10			0.486	0.492	0.473
	5	none	1	lbfgs	none	10			0487	0489	0.473
	6	elasticnet	1	saga	0.3	15			0.481	0.481	0.467
	7	L1	6	lbfgs	none	15			0.481	0.482	0.467
Decision Tree		Criterion	Max_depth	Min_samples_split	Min_samples_leaf	splitter	Max_feature	Num_of_variables			
	1	gini	2	1000	500	best	none	10	0.488	0.484	0.459 underfitting
	2	gini	5	1000	500	best	none	5	0.515	0.510	0.490
	3	gini	5	500	250	best	none	10	0.512	0.514	0.491
	4	entropy	5	1000	500	best	none	10	0.521	0.516	0.497
	5	entropy	10	1000	500	best	none	10	0.525	0.520	0.500
	6	entropy	20	1000	500	best	none	10	0.528	0.527	0.504
	7	entropy	100	500	250	best	None	10	0.525	0.517	0.495
	8	entropy	50	500	250	best	5	15	0.518	0.505	0.488
	9	entropy	50	500	250	random	5	15	0.331	0.331	0.296 underfitting
Random Forest		N_estimator	Max_depth	Min_sample_split	Min_samples_leaf	max_feature	criterion	Num_of_variables			
	1	3	2	1000	500	5	gini	5	0.467	0.460	0.449 underfitting
	2	10	2	1000	500	8	gini	10	0.480	0.479	0.466
	3	10	20	1000	500	8	gini	10	0.527	0.519	0.501
	4	100	20	1000	500	8	gini	15	0.525	0.521	0.500
	5	100	20	1000	500	8	entropy	10	0.525	0.530	0.504
	6	50	20	2000	1000	10	entropy	10	0.527	0.524	0.505
	7	50	20	none	none	10	entropy	15	0.542	0.520	0.499 Overfitting
Catboost		Boosting_type	Max_depth	Iterations	Learning rate	Num of variables					
	1	none	2	5	none	5			0.491	0.499	0.476 underfitting
	2	none	16	20	none	10			0.529	0.520	0.505
	3	Plain	16	20	0.2	10			0.526	0.528	0.504
	4	Ordered	16	20	0.2	10			0.524	0.523	0.500
	5	plain	16	30	none	10			0.530	0.520	0.505
	6	plain	16	30	none	15			0.529	0.523	0.503
XGBoost		Max_depth	gamma	n_estimators	Learning rate	Colsample_bytree	Num_of_variables				
	1	2	0	5	0.01	0.5	10		0.501	0.500	0.477
	2	10	0	20	0.05	0.5	10		0.520	0.521	0.499
	3	10	0	20	0.05	0.5	5		0.505	0.508	0.483
	4	10	0.5	20	0.2	0.5	10		0.525	0.533	0.502
	5	10	1	20	0.3	0.8	10		0.534	0.522	0.505
	6	10	3	30	0.3	1	10		0.526	0.532	0.507
	7	10	3	30	0.3	1	15		0.530	0.525	0.507
Neural Network		activation	solver	Learning rate	Hidden_layer_sizes	Num_of_variables					
	1	relu	adam	constant	2	10			0.502	0.508	0.487
	2	relu	adam	constant	1,3	10			0.248	0.249	0.237 underfitting
	3	identity	adam	constant	5	10			0.489	0.487	0.475
	4	identiy	sgd	adpative	2	10			0.496	0.487	0.479 overfitting
	5	identity	sgd	adaptive	2	15			0.483	0.485	0.470
	6	Identity	sgd	adaptive	2	5			0.480	0.476	0.465

For modeling part, I tried six models including Logistics Regression, decision tree, random forest, Catboost, XGBoost, and Neural network. The table above shows different hyperparameter parameters for each model.

There are few observations while trying different hyperparameters:

1. **Logistic regression**, as a typical base-line model, generally perform worse than other complex models.
2. When the splitter is set as random, **Decision Tree** shows to be underfitting. The FDR is around 0.331 for training, testing, and OOT, almost 0.2 worse than model with other hypermeters. Also, when the max-depth is too small, decision tree tends to be underfitting.
3. **Random Forest**, like dehcision tree, is underfitting when the max-depth and number of estimators are set at a very small number, and when the numbers are large, the model will be overfitting.
4. **Catboost**, like other tree models, is underfitting when max-depth is very small(2 at this case). The model will also be underfitting when the iteration is small (5) and number of variables is at 5.
5. **XGBoost** is very flexible according to the table above, the performance does not change responding to the change very much of hyperparameters.
6. **Neural Network** generally perform worse than other models. It showed to be overfitting when the hidden layer is set at 2 and learning rate is adaptive.

I chose random forest as my final model since it is very fast (2 minutes), compared to other models that has similar performance, and at the same time achieve 0.524 FDR for test set and 0.504 for OOT.

6. Summary of results. The 3 results tables (trn, tst, oot) for your final model. Description of result

   In the final **Random Forest model**, I included 10 variables:
   1. 'fulladdress_day_since'
   2. 'name_dob_count_30'
   3. 'address_unique_count_for_name_homephone_60'
   4. 'fulladdress_unique_count_for_dob_homephone_3'
   5. 'address_unique_count_for_homephone_name_dob_30'
   6. 'address_unique_count_for_ssn_name_dob_14'
   7. 'address_day_since'
   8. 'address_count_14'
   9. 'address_count_7'
   10. 'address_count_0_by_30'.

   The **hyperparameter setting** is:
   criterion = 'entropy', n_estimators=50, max_depth=20, max_features=10, min_samples_split= 2000,min_samples_leaf = 1000

The **Result** is

Traning	0.527
Testing	0.524
Out of Time	0.504

I found max-depth and max_feature important hyperparmeters for random forest, for its influences on whether model will be overfitting or underfitting.

Top 5 Variables according to feature importance

* The ranking list is similar to the rank of variables from the wrapper

1) fulladdress_unique_count_for_dob_homephone_3 0.447270
2) name_dob_count_30 0.433061
3) address_count_14 0.051045
4) fulladdress_day_since 0.037391
5) address_count_0_by_30 0.013477

Training Set

Traning		#Records		#Goods		#Bads		#Fraud Rate				
		583454		575030		8424		0.01443816				
			Bin Statistics				Culmulative	Statistics				
bin	#recs	#g	#b	%g	%b	tot	cg	cb	%cg	FDR	KS	FPR
1	5835	1729	4106	29.6315338	70.3684662	5835	1729	4106	0.30	48.74	48.44	0.42
2	5834	5623	211	96.3832705	3.61672952	11669	7352	4317	1.28	51.25	49.97	1.70
3	5835	5744	91	98.4404456	1.55955441	17504	13096	4408	2.28	52.33	50.05	2.97
4	5834	5766	68	98.8344189	1.16558108	23338	18862	4476	3.28	53.13	49.85	4.21
5	5835	5781	54	99.0745501	0.92544987	29173	24643	4530	4.29	53.77	49.49	5.44
6	5834	5795	39	99.331505	0.66849503	35007	30438	4569	5.29	54.24	48.94	6.66
7	5835	5794	41	99.2973436	0.70265638	40842	36232	4610	6.30	54.72	48.42	7.86
8	5834	5809	25	99.5714775	0.42852245	46676	42041	4635	7.31	55.02	47.71	9.07
9	5835	5805	30	99.4858612	0.51413882	52511	47846	4665	8.32	55.38	47.06	10.26
10	5834	5796	38	99.3486459	0.65135413	58345	53642	4703	9.33	55.83	46.50	11.41
11	5835	5795	40	99.3144816	0.68551842	64180	59437	4743	10.34	56.30	45.97	12.53
12	5834	5798	36	99.3829277	0.61707233	70014	65235	4779	11.34	56.73	45.39	13.65
13	5835	5800	35	99.4001714	0.59982862	75849	71035	4814	12.35	57.15	44.79	14.76
14	5835	5792	43	99.2630677	0.73693231	81684	76827	4857	13.36	57.66	44.30	15.82
15	5834	5793	41	99.2972232	0.70277683	87518	82620	4898	14.37	58.14	43.78	16.87
16	5835	5798	37	99.3658955	0.63410454	93353	88418	4935	15.38	58.58	43.21	17.92
17	5834	5788	46	99.2115187	0.78848132	99187	94206	4981	16.38	59.13	42.75	18.91
18	5835	5801	34	99.4173093	0.58269066	105022	100007	5015	17.39	59.53	42.14	19.94
19	5834	5783	51	99.1258142	0.87418581	110856	105790	5066	18.40	60.14	41.74	20.88
20	5835	5794	41	99.2973436	0.70265638	116691	111584	5107	19.40	60.62	41.22	21.85

Test Set

Testing	# Records		#Goods		#Bads		#Fraud Rate					
	250053		246470		3583		0.014328962					
	Bin Statistics					Culmulative Statistics						FPR
Population bin %	#Records	#goods	#bads	%goods	%bads	Total of records	Cumulative Goods	Cumulative Bads	%Goods	% Bads	KS	0.40584598
1	2501	722	1779	28.87	71.13	2501	722	1779	0.29	49.65	49.36	1.6858217
2	2500	2417	83	96.68	3.32	5001	3139	1862	1.27	51.97	50.69	2.92980618
3	2501	2454	47	98.12	1.88	7502	5593	1909	2.27	53.28	51.01	4.19584416
4	2500	2484	16	99.36	0.64	10002	8077	1925	3.28	53.73	50.45	5.40850846
5	2501	2475	26	98.96	1.04	12503	10552	1951	4.28	54.45	50.17	6.63123093
6	2500	2485	15	99.40	0.60	15003	13037	1966	5.29	54.87	49.58	7.81813602
7	2501	2482	19	99.24	0.76	17504	15519	1985	6.30	55.40	49.10	8.98701947
8	2500	2482	18	99.28	0.72	20004	18001	2003	7.30	55.90	48.60	10.1687345
9	2501	2489	12	99.52	0.48	22505	20490	2015	8.31	56.24	47.92	11.2995573
10	2500	2482	18	99.28	0.72	25005	22972	2033	9.32	56.74	47.42	12.4437928
11	2501	2488	13	99.48	0.52	27506	25460	2046	10.33	57.10	46.77	13.5518914
12	2500	2484	16	99.36	0.64	30006	27944	2062	11.34	57.55	46.21	14.5834132
13	2501	2477	24	99.04	0.96	32507	30421	2086	12.34	58.22	45.88	15.5596026
14	2500	2472	28	98.88	1.12	35007	32893	2114	13.35	59.00	45.66	16.6508235
15	2501	2490	11	99.56	0.44	37508	35383	2125	14.36	59.31	44.95	17.6778711
16	2500	2483	17	99.32	0.68	40008	37866	2142	15.36	59.78	44.42	18.6800926
17	2501	2483	18	99.28	0.72	42509	40349	2160	16.37	60.28	43.91	19.646789
18	2501	2481	20	99.20	0.80	45010	42830	2180	17.38	60.84	43.47	20.6347905
19	2500	2484	16	99.36	0.64	47510	45314	2196	18.39	61.29	42.90	21.5987347
20	2501	2484	17	99.32	0.68	50011	47798	2213	19.39	61.76	42.37	

OOT

OOT		#Records		#Goods		#Bads		#Fraud Rate				
		166493		164107.00		2386		0.01433093				
			Bin Statistics				Culmulative	Statistics				
bin	#recs	#g	#b	%g	%b	tot	cg	cb	%cg	FDR	KS	FPR
1	1665	542	1123	32.55	67.45	1665	542	1123	0.3	47.1	46.7	0.5
2	1665	1606	59	96.46	3.54	3330	2148	1182	1.3	49.5	48.2	1.8
3	1665	1641	24	98.56	1.44	4995	3789	1206	2.3	50.5	48.2	3.1
4	1665	1647	18	98.92	1.08	6660	5436	1224	3.3	51.3	48.0	4.4
5	1665	1659	6	99.64	0.36	8325	7095	1230	4.3	51.6	47.2	5.8
6	1665	1651	14	99.16	0.84	9990	8746	1244	5.3	52.1	46.8	7.0
7	1665	1651	14	99.16	0.84	11655	10397	1258	6.3	52.7	46.4	8.3
8	1664	1649	15	99.10	0.90	13319	12046	1273	7.3	53.4	46.0	9.5
9	1665	1661	4	99.76	0.24	14984	13707	1277	8.4	53.5	45.2	10.7
10	1665	1652	13	99.22	0.78	16649	15359	1290	9.4	54.1	44.7	11.9
11	1665	1649	16	99.04	0.96	18314	17008	1306	10.4	54.7	44.4	13.0
12	1665	1652	13	99.22	0.78	19979	18660	1319	11.4	55.3	43.9	14.1
13	1665	1654	11	99.34	0.66	21644	20314	1330	12.4	55.7	43.4	15.3
14	1665	1657	8	99.52	0.48	23309	21971	1338	13.4	56.1	42.7	16.4
15	1665	1656	9	99.46	0.54	24974	23627	1347	14.4	56.5	42.1	17.5
16	1665	1651	14	99.16	0.84	26639	25278	1361	15.4	57.0	41.6	18.6
17	1665	1654	11	99.34	0.66	28304	26932	1372	16.4	57.5	41.1	19.6
18	1665	1654	11	99.34	0.66	29969	28586	1383	17.4	58.0	40.5	20.7
19	1665	1655	10	99.40	0.60	31634	30241	1393	18.4	58.4	40.0	21.7
20	1665	1654	11	99.34	0.66	33299	31895	1404	19.4	58.8	39.4	22.7

Conclusion:

The OOT (0.504) shows that the Random Forest model can eliminate about 50% of the fraud by declining only about 3% of the applications.