Exploring Big Data

Project Description

This task is based on a synthesised transaction dataset containing 3 months' worth of transactions for 100 hypothetical customers. It contains purchases, recurring transactions, and salary transactions.

This task will assess some of the core skills expected in a Big Data engineer at ANZ, particularly your familiarity with Apache Spark.

Using each API, perform the following transformation steps using the synthetic transaction file as input referenced as an input argument to your program. Output the results to a local file.

- Project only the records where status=authorized AND card_present_flag=0
- Split the long_lat and merchant_long_lat fields into long, lat and merch_long, merch_lat fields
- Output the data as a CSV file

Import Libraries

In [1]:

```
import numpy as np
from numpy import count_nonzero, median, mean
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random

# Plotly
# import plotly.express as px
# import plotly.offline as py
# import plotly.graph_objs as go

import sweetviz

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
# import researchpy as rp

import datetime
from datetime import datetime, timedelta

# import eli5
# from IPython.display import display

#import os
#import zipfile
import scipy.stats
from collections import Counter

import sklearn
# from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder, OneHotEn
# from sklearn.linear_model import LinearRegression, LogisticRegression, ElasticNet, Las
# from sklearn.model_selection import cross_val_score, train_test_split
```

```
# from sklearn.metrics import accuracy_score, auc, classification_report, confusion_matr
# from sklearn.metrics import plot_confusion_matrix, plot_roc_curve

# from sklearn.linear_model import ElasticNet, Lasso, LinearRegression, LogisticRegressi
# from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, ExtraTreeClass
# from sklearn.svm import SVC, SVR, LinearSVC, LinearSVR
# from sklearn.naive_bayes import GaussianNB, MultinomialNB

%matplotlib inline
#sets the default autosave frequency in seconds
%autosave 60
sns.set_style('dark')
sns.set(font_scale=1.2)

plt.rc('axes', titlesize=9)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=12)
plt.rc('ytick', labelsize=12)

import warnings
warnings.filterwarnings('ignore')

# Use Feature-Engine library
#import feature_engine
#from feature_engine import imputation as mdi
#from feature_engine.outlier_removers import Winsorizer
#from feature_engine import categorical_encoders as ce
#from feature_engine.discretisation import EqualWidthDiscretiser, EqualFrequencyDiscreti
#from feature_engine.discretisation import ArbitraryDiscretiser, DecisionTreeDiscretiser
#from feature_engine.encoding import OrdinalEncoder

pd.set_option('display.max_columns',None)
#pd.set_option('display.max_rows',None)
pd.set_option('display.width', 1000)
pd.set_option('display.float_format','{:.2f}'.format)

random.seed(0)
np.random.seed(0)
np.set_printoptions(suppress=True)
```

```
Autosaving every 60 seconds
```

Exploratory Data Analysis

In [2]:
```
df = pd.read_csv("BIGDATAContentTask.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description	merchant_id
0	authorized	1.00	NaN	ACC-1598451071	AUD	153.41 -27.95	POS	81c48296-73be-44a7-befa-d053f48ce7cd
1	authorized	0.00	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	830a451c-316e-4a6a-bf25-e37caedca49e
2	authorized	1.00	NaN	ACC-1222300524	AUD	151.23 -33.94	POS	835c231d-8cdf-4e96-859d-e9d571760cf0

3	authorized	1.00	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-POS	48514682-c78a-4a88-b0da-2d6302e64673
4	authorized	1.00	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	b4e02c10-0852-4273-b8fd-7b3395e32eb0

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12043 entries, 0 to 12042
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   status             12043 non-null  object 
 1   card_present_flag  7717 non-null   float64
 2   bpay_biller_code   885 non-null    object 
 3   account            12043 non-null  object 
 4   currency           12043 non-null  object 
 5   long_lat           12043 non-null  object 
 6   txn_description    12043 non-null  object 
 7   merchant_id        7717 non-null   object 
 8   merchant_code      883 non-null    float64
 9   first_name         12043 non-null  object 
 10  balance            12043 non-null  float64
 11  date               12043 non-null  object 
 12  gender             12043 non-null  object 
 13  age                12043 non-null  int64  
 14  merchant_suburb    7717 non-null   object 
 15  merchant_state     7717 non-null   object 
 16  extraction         12043 non-null  object 
 17  amount             12043 non-null  float64
 18  transaction_id     12043 non-null  object 
 19  country            12043 non-null  object 
 20  customer_id        12043 non-null  object 
 21  merchant_long_lat  7717 non-null   object 
 22  movement           12043 non-null  object 
dtypes: float64(4), int64(1), object(18)
memory usage: 2.1+ MB
```

In [5]:
```
df.describe()
```

Out[5]:

	card_present_flag	merchant_code	balance	age	amount
count	7717.00	883.00	12043.00	12043.00	12043.00
mean	0.80	0.00	14704.20	30.58	187.93
std	0.40	0.00	31503.72	10.05	592.60
min	0.00	0.00	0.24	18.00	0.10
25%	1.00	0.00	3158.59	22.00	16.00
50%	1.00	0.00	6432.01	28.00	29.00
75%	1.00	0.00	12465.94	38.00	53.66
max	1.00	0.00	267128.52	78.00	8835.98

In [6]:
```
df.columns
```

Out[6]: Index(['status', 'card_present_flag', 'bpay_biller_code', 'account', 'currency', 'long_lat', 'txn_description', 'merchant_id', 'merchant_code', 'first_name', 'balance', 'date', 'gender', 'age', 'merchant_suburb', 'merchant_state', 'extraction', 'amount', 'transaction_id', 'country', 'customer_id', 'merchant_long_lat', 'movement'], dtype='object')

In [7]:
```python
df.status.value_counts()
```

Out[7]:
```
authorized    7717
posted        4326
Name: status, dtype: int64
```

In [8]:
```python
df2 = df[df.status == "authorized"]
df2
```

Out[8]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description	merchan
0	authorized	1.00	NaN	ACC-1598451071	AUD	153.41 -27.95	POS	81c482 73be-44 b d053f48ce
1	authorized	0.00	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	830a45 316e-4a b e37caedca
2	authorized	1.00	NaN	ACC-1222300524	AUD	151.23 -33.94	POS	835c23 8cdf-4e 85 e9d57176
3	authorized	1.00	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-POS	485146 c78a-4a b0 2d6302e64
4	authorized	1.00	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	b4e02c 0852-42 b 7b3395e32
...	...	...	...	...	...	...	...	
12038	authorized	0.00	NaN	ACC-3021093232	AUD	149.83 -29.47	POS	32aa73 b7c2-41 b1 6271b96ce
12039	authorized	1.00	NaN	ACC-1608363396	AUD	151.22 -33.87	SALES-POS	296a05 8552-48 ac ec37065b5
12040	authorized	1.00	NaN	ACC-3827517394	AUD	151.12 -33.89	POS	e5975a 08f7-47 a3 24cc0e35e
12041	authorized	1.00	NaN	ACC-2920611728	AUD	144.96 -37.76	SALES-POS	af4905 591d-4b bc 27730b70e
12042	authorized	1.00	NaN	ACC-1443681913	AUD	150.92 -33.77	SALES-POS	f31f4b 2040-40 a1 b141bb274

7717 rows × 23 columns

In [9]:
```
df3 = df2[df2.card_present_flag == 0.00]
df3
```

Out[9]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description	merchant
1	authorized	0.00	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	830a45 316e-4a bf e37caedca
21	authorized	0.00	NaN	ACC-2890243754	AUD	153.32 -27.93	POS	7e8bf6 e724-43 a4 3538a3e27
23	authorized	0.00	NaN	ACC-2615038700	AUD	145.35 -38.03	POS	354f40 55bc-4a a0 c7faede29
29	authorized	0.00	NaN	ACC-1710017148	AUD	150.82 -34.01	SALES-POS	4af250 a1a4-46 90 240d790e5
31	authorized	0.00	NaN	ACC-3485804958	AUD	138.52 -35.01	POS	a08935 99a8-49 b7 f8de51ca0
...	...	...	...	...	...	...	...	
12012	authorized	0.00	NaN	ACC-3954677887	AUD	115.72 -32.28	SALES-POS	4995bf 13d8-4 a9 d5331da83
12015	authorized	0.00	NaN	ACC-1598451071	AUD	153.41 -27.95	POS	e4758c e8d8-49 99 a823a86dc
12016	authorized	0.00	NaN	ACC-2249586092	AUD	115.98 -32.07	SALES-POS	23eccb 684e-43 b9 dc307517c
12031	authorized	0.00	NaN	ACC-1443681913	AUD	150.92 -33.77	SALES-POS	6fcdc9 3548-40 a2 9dbc6cb64
12038	authorized	0.00	NaN	ACC-3021093232	AUD	149.83 -29.47	POS	32aa73 b7c2-41 b1 6271b96ce

1523 rows × 23 columns

In [10]:
```
df3.reset_index(inplace=True, drop=True)
```

In [11]:
```
df3
```

Out[11]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description	merchant_
0	authorized	0.00	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	830a451 316e-4a6 bf2 e37caedca4
1	authorized	0.00	NaN	ACC-2890243754	AUD	153.32 -27.93	POS	7e8bf66 e724-435 a40 3538a3e27b
2	authorized	0.00	NaN	ACC-2615038700	AUD	145.35 -38.03	POS	354f40c 55bc-4a8 a00 c7faede29f
3	authorized	0.00	NaN	ACC-1710017148	AUD	150.82 -34.01	SALES-POS	4af2504 a1a4-468 90b 240d790e53
4	authorized	0.00	NaN	ACC-3485804958	AUD	138.52 -35.01	POS	a08935a 99a8-49f b73 f8de51ca0a
...	...	...	...	...	...	...	...	
1518	authorized	0.00	NaN	ACC-3954677887	AUD	115.72 -32.28	SALES-POS	4995bfd 13d8-4c5 a9f d5331da83f
1519	authorized	0.00	NaN	ACC-1598451071	AUD	153.41 -27.95	POS	e4758c3 e8d8-49b 990 a823a86dca
1520	authorized	0.00	NaN	ACC-2249586092	AUD	115.98 -32.07	SALES-POS	23eccb6 684e-432 b95 dc307517c8
1521	authorized	0.00	NaN	ACC-1443681913	AUD	150.92 -33.77	SALES-POS	6fcdc95 3548-40b a2c 9dbc6cb64e
1522	authorized	0.00	NaN	ACC-3021093232	AUD	149.83 -29.47	POS	32aa73d b7c2-416 b14 6271b96ce7

1523 rows × 23 columns

Data Visualization

Univariate Data Exploration

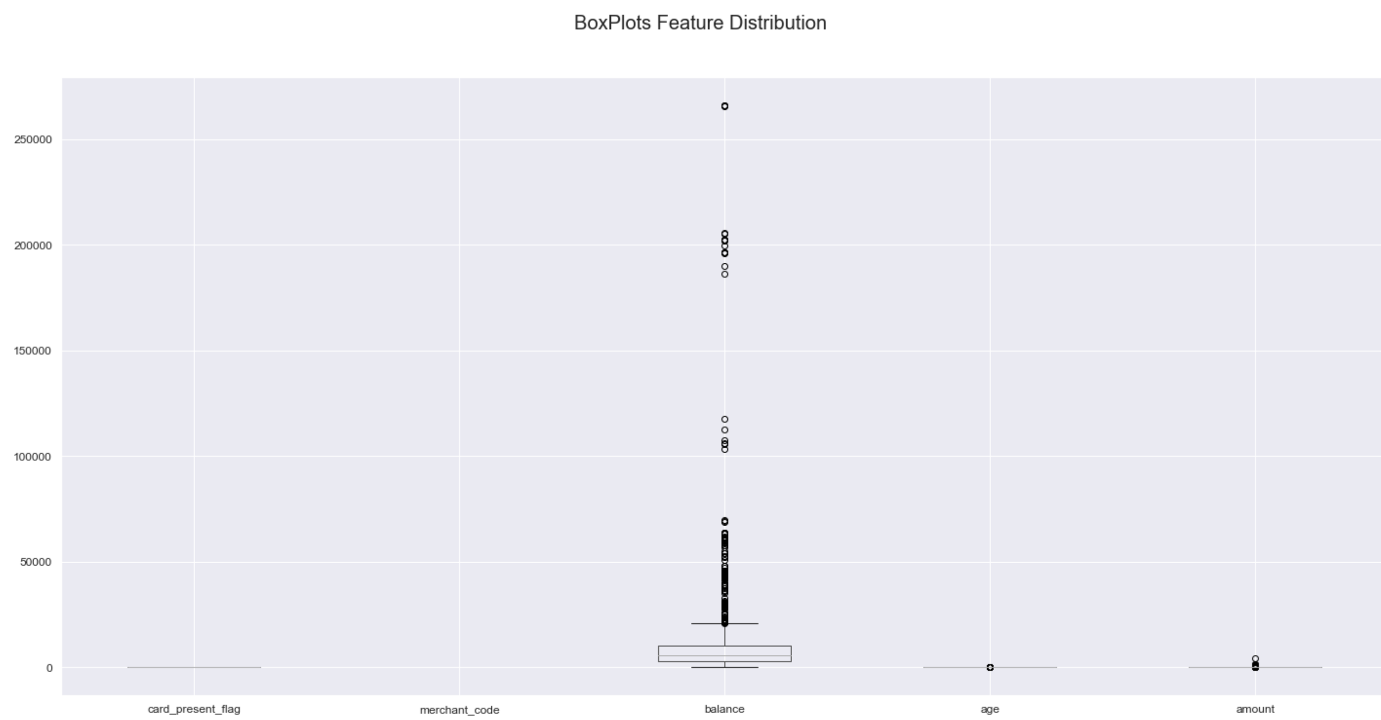
In [12]:

```
df3.hist(bins=50, figsize=(20,10))
plt.suptitle('Histogram Feature Distribution', x=0.5, y=1.02, ha='center', fontsize=20)
```

```python
plt.tight_layout()
plt.show()
```

In [13]:
```python
df3.boxplot(figsize=(20,10))
plt.suptitle('BoxPlots Feature Distribution', x=0.5, y=1.02, ha='center', fontsize=20)

plt.tight_layout()
plt.show()
```



In [14]:
```python
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1523 entries, 0 to 1522
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   status             1523 non-null   object 
 1   card_present_flag  1523 non-null   float64
 2   bpay_biller_code   0 non-null      object 
```

```
 3   account            1523 non-null   object
 4   currency           1523 non-null   object
 5   long_lat           1523 non-null   object
 6   txn_description    1523 non-null   object
 7   merchant_id        1523 non-null   object
 8   merchant_code      0 non-null      float64
 9   first_name         1523 non-null   object
 10  balance            1523 non-null   float64
 11  date               1523 non-null   object
 12  gender             1523 non-null   object
 13  age                1523 non-null   int64
 14  merchant_suburb    1523 non-null   object
 15  merchant_state     1523 non-null   object
 16  extraction         1523 non-null   object
 17  amount             1523 non-null   float64
 18  transaction_id     1523 non-null   object
 19  country            1523 non-null   object
 20  customer_id        1523 non-null   object
 21  merchant_long_lat  1523 non-null   object
 22  movement           1523 non-null   object
dtypes: float64(4), int64(1), object(18)
memory usage: 273.8+ KB
```

Data Preprocessing

Feature Engineering

In [15]:
```
df3.columns
```

Out[15]:
```
Index(['status', 'card_present_flag', 'bpay_biller_code', 'account', 'currency', 'long_l
at', 'txn_description', 'merchant_id', 'merchant_code', 'first_name', 'balance', 'date',
'gender', 'age', 'merchant_suburb', 'merchant_state', 'extraction', 'amount', 'transacti
on_id', 'country', 'customer_id', 'merchant_long_lat', 'movement'], dtype='object')
```

In [16]:
```
df3.drop(['status', 'card_present_flag', 'bpay_biller_code', 'account', 'currency', 'lon
           'txn_description', 'merchant_id', 'merchant_code', 'first_name'], inplace=True
```

In [17]:
```
df3.head()
```

Out[17]:

	balance	date	gender	age	merchant_suburb	merchant_state	extraction	amount	
0	21.20	1/8/2018	F	26	Sydney	NSW	2018-08-01T01:13:45.000+0000	14.19	13270a2a90
1	275.93	1/8/2018	M	37	Lismore	NSW	2018-08-01T08:19:14.000+0000	24.77	1f12467d33
2	30583.15	1/8/2018	F	43	Mordialloc	VIC	2018-08-01T08:47:48.000+0000	12.08	49417bad3
3	1625.34	1/8/2018	F	19	Alexandria	NSW	2018-08-01T09:11:00.000+0000	11.57	82acf0379
4	12529.59	1/8/2018	F	34	Findon	SA	2018-08-01T09:19:06.000+0000	33.89	89050ee5c

In [18]:
```
df3.drop(['date', 'merchant_suburb', 'extraction', 'transaction_id', 'country', 'custome
```

In [19]:
```
df3.head()
```

Out[19]:

	balance	gender	age	merchant_state	amount	movement

0	21.20	F	26	NSW	14.19	debit
1	275.93	M	37	NSW	24.77	debit
2	30583.15	F	43	VIC	12.08	debit
3	1625.34	F	19	NSW	11.57	debit
4	12529.59	F	34	SA	33.89	debit

In [ ]:

Save to CSV

In [20]:
```
df3.to_csv("final.csv", index=False)
```

Regression Analysis

Logistic Regression (StatsModel)

In [21]:
```
df3.columns
```

Out[21]:
```
Index(['balance', 'gender', 'age', 'merchant_state', 'amount', 'movement'], dtype='objec
t')
```

In [22]:
```
df3.movement.value_counts()
```

Out[22]:
```
debit    1523
Name: movement, dtype: int64
```

In [23]:
```
df4 = pd.get_dummies(data=df3)
```

In [24]:
```
df4
```

Out[24]:

	balance	age	amount	gender_F	gender_M	merchant_state_ACT	merchant_state_NSW	merchant_state_NT
0	21.20	26	14.19	1	0	0	1	0
1	275.93	37	24.77	0	1	0	1	0
2	30583.15	43	12.08	1	0	0	0	0
3	1625.34	19	11.57	1	0	0	1	0
4	12529.59	34	33.89	1	0	0	0	0
...	...	...	...	...	...	...	...	...
1518	9901.03	47	15.91	1	0	0	0	0
1519	2194.26	26	25.88	1	0	0	0	0
1520	12963.75	19	9.90	0	1	0	0	0
1521	5540.27	31	70.51	0	1	0	1	0
1522	14054.14	30	9.79	1	0	0	0	0

1523 rows × 14 columns

In [25]:
```
df4.columns
```

Out[25]: Index(['balance', 'age', 'amount', 'gender_F', 'gender_M', 'merchant_state_ACT', 'merchant_state_NSW', 'merchant_state_NT', 'merchant_state_QLD', 'merchant_state_SA', 'merchant_state_TAS', 'merchant_state_VIC', 'merchant_state_WA', 'movement_debit'], dtype='object')

In [26]:
```
y = df4['movement_debit']
X = df4[['balance', 'age', 'amount', 'gender_F', 'gender_M', 'merchant_state_ACT', 'merc
```

In [27]:
```
X = sm.add_constant(X)
```

In [28]:
```
model = sm.Logit(y, X).fit()
```

```
---------------------------------------------------------------------------
PerfectSeparationError                    Traceback (most recent call last)
Input In [28], in <cell line: 1>()
----> 1 model = sm.Logit(y, X).fit()

File C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py:1983, in Logit.fit(self, start_params, method, maxiter, full_output, disp, callback, **kwargs)
   1980 @Appender(DiscreteModel.fit.__doc__)
   1981 def fit(self, start_params=None, method='newton', maxiter=35,
   1982         full_output=1, disp=1, callback=None, **kwargs):
-> 1983     bnryfit = super().fit(start_params=start_params,
   1984                           method=method,
   1985                           maxiter=maxiter,
   1986                           full_output=full_output,
   1987                           disp=disp,
   1988                           callback=callback,
   1989                           **kwargs)
   1991     discretefit = LogitResults(self, bnryfit)
   1992     return BinaryResultsWrapper(discretefit)

File C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py:230, in DiscreteModel.fit(self, start_params, method, maxiter, full_output, disp, callback, **kwargs)
    227 else:
    228     pass  # TODO: make a function factory to have multiple call-backs
--> 230 mlefit = super().fit(start_params=start_params,
    231                      method=method,
    232                      maxiter=maxiter,
    233                      full_output=full_output,
    234                      disp=disp,
    235                      callback=callback,
    236                      **kwargs)
    238 return mlefit

File C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\model.py:563, in LikelihoodModel.fit(self, start_params, method, maxiter, full_output, disp, fargs, callback, retall, skip_hessian, **kwargs)
    560     del kwargs["use_t"]
    562 optimizer = Optimizer()
--> 563 xopt, retvals, optim_settings = optimizer._fit(f, score, start_params,
    564                                                fargs, kwargs,
    565                                                hessian=hess,
    566                                                method=method,
    567                                                disp=disp,
    568                                                maxiter=maxiter,
    569                                                callback=callback,
    570                                                retall=retall,
    571                                                full_output=full_output)
    572 # Restore cov_type, cov_kwds and use_t
    573 optim_settings.update(kwds)

File C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\optimizer.py:241, in Op
```

```
timizer._fit(self, objective, gradient, start_params, fargs, kwargs, hessian, method, ma
xiter, full_output, disp, callback, retall)
    238     fit_funcs.update(extra_fit_funcs)
    240 func = fit_funcs[method]
--> 241 xopt, retvals = func(objective, gradient, start_params, fargs, kwargs,
    242                      disp=disp, maxiter=maxiter, callback=callback,
    243                      retall=retall, full_output=full_output,
    244                      hess=hessian)
    246 optim_settings = {'optimizer': method, 'start_params': start_params,
    247                   'maxiter': maxiter, 'full_output': full_output,
    248                   'disp': disp, 'fargs': fargs, 'callback': callback,
    249                   'retall': retall, "extra_fit_funcs": extra_fit_funcs}
    250 optim_settings.update(kwargs)

File C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\optimizer.py:443, in _f
it_newton(f, score, start_params, fargs, kwargs, disp, maxiter, callback, retall, full_o
utput, hess, ridge_factor)
    441         history.append(newparams)
    442     if callback is not None:
--> 443         callback(newparams)
    444     iterations += 1
    445 fval = f(newparams, *fargs)  # this is the negative likelihood

File C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py:2
14, in DiscreteModel._check_perfect_pred(self, params, *args)
    211 if (self.raise_on_perfect_prediction and
    212         np.allclose(fittedvalues - endog, 0)):
    213     msg = "Perfect separation detected, results not available"
--> 214     raise PerfectSeparationError(msg)

PerfectSeparationError: Perfect separation detected, results not available
```

In [ ]:

```
model.summary()
```

In [ ]:

```
logitfit = smf.logit(formula = 'DF ~ Debt_Service_Coverage + cash_security_to_curLiab +
```

In [ ]:

```
logitfit = smf.logit(formula = 'DF ~ TNW + C(seg2)', data = hgcdev).fit()
```

In [ ]:

```
```