Insurance companies invest a lot of time and money into optimizing their pricing and accurately estimating the likelihood that customers will make a claim. In many countries insurance it is a legal requirement to have car insurance in order to drive a vehicle on public roads, so the market is very large!

(`Source: https://www.accenture.com/_acnmedia/pdf-84/accenture-machine-leaning-insurance.pdf`)

Knowing all of this, On the Road car insurance have requested your services in building a model to predict whether a customer will make a claim on their insurance during the policy period. As they have very little expertise and infrastructure for deploying and monitoring machine learning models, they've asked you to identify the single feature that results in the best performing model, as measured by accuracy, so they can start with a simple model in production.

They have supplied you with their customer data as a csv file called `car_insurance.csv`, along with a table detailing the column names and descriptions below.

The dataset

Column	Description
`id`	Unique client identifier
`age`	Client's age: • `0` : 16-25 • `1` : 26-39 • `2` : 40-64 • `3` : 65+
`gender`	Client's gender: • `0` : Female • `1` : Male
`driving_experience`	Years the client has been driving: • `0` : 0-9 • `1` : 10-19 • `2` : 20-29 • `3` : 30+
`education`	Client's level of education: • `0` : No education • `1` : High school • `2` : University
`income`	Client's income level: • `0` : Poverty • `1` : Working class • `2` : Middle class • `3` : Upper class
`credit_score`	Client's credit score (between zero and one)
`vehicle_ownership`	Client's vehicle ownership status: • `0` : Does not own their vehilce (paying off finance) • `1` : Owns their vehicle
`vehcile_year`	Year of vehicle registration: • `0` : Before 2015 • `1` : 2015 or later
`married`	Client's marital status: • `0` : Not married • `1` : Married
`children`	Client's number of children
`postal_code`	Client's postal code
`annual_mileage`	Number of miles driven by the client each year
`vehicle_type`	Type of car: • `0` : Sedan • `1` : Sports car
`speeding_violations`	Total number of speeding violations received by the client
`duis`	Number of times the client has been caught driving under the influence of alcohol
`past_accidents`	Total number of previous accidents the client has been involved in
`outcome`	Whether the client made a claim on their car insurance (response variable): • `0` : No claim • `1` : Made a claim

```python
# Import required modules
import pandas as pd
import numpy as np
from statsmodels.formula.api import logit

# Start coding!
data = pd.read_csv('car_insurance.csv')
data.isna().any()
data['annual_mileage'].isna().sum()
data['credit_score'].isna().sum()
data.head()
```

•••	•••	•••	•••	driving_experie...	ed...	income	credi...	vehicle_owne...	ve
0	569520	3	0	0-9y	high school	upper class	0.6290273139	1	af
1	750365	0	1	0-9y	none	poverty	0.357757117	0	be
2	199901	0	0	0-9y	high school	working class	0.4931457852	1	be
3	478866	0	1	0-9y	university	working class	0.2060128507	1	be
4	731664	1	1	10-19y	none	working class	0.3883658882	1	be

Rows: 5 Expand

```python
#driving experience
data['driving_experience'] = data['driving_experience'].astype('category').cat.codes
mdl1 = logit('outcome ~ driving_experience', data = data).fit()
print(mdl1.params)
exp1 = pd.DataFrame({'driving_experience':np.arange(0, 3, 1)})
pre1 = exp1.assign(outcome = mdl1.predict(exp1))
c1 = mdl1.pred_table()
TN1 = c1[0,0]
TP1 = c1[1,1]
FN1 = c1[1,0]
FP1 = c1[0,1]
acc1 = (TN1 + TP1) / (TN1 + TP1 + FN1 + FP1)
print("accuracy: ", acc1)
```

```
Optimization terminated successfully.
         Current function value: 0.467390
         Iterations 7
Intercept             0.515887
driving_experience   -1.668485
dtype: float64
accuracy:  0.7771
```

```python
#education
data['education'] = data['education'].astype('category').cat.codes
mdl2 = logit('outcome ~ education', data = data).fit()
print(mdl2.params)
exp2 = pd.DataFrame({'education':np.arange(0, 2, 1)})
pre2 = exp2.assign(outcome = mdl2.predict(exp2))
c2 = mdl2.pred_table()
TN2 = c2[0,0]
TP2 = c2[1,1]
FN2 = c2[1,0]
FP2 = c2[0,1]
acc2 = (TN2 + TP2) / (TN2 + TP2 + FN2 + FP2)
print("accuracy: ", acc2)
```

```
Optimization terminated successfully.
         Current function value: 0.617408
         Iterations 5
Intercept   -0.573689
education   -0.223746
dtype: float64
accuracy:  0.6867
```

```
#income
data['income'] = data['income'].astype('category').cat.codes
mdl3 = logit('outcome ~ income', data = data).fit()
print(mdl3.params)
exp3 = pd.DataFrame({'income':np.arange(0, 3, 1)})
pre3 = exp3.assign(outcome = mdl3.predict(exp3))
c3 = mdl3.pred_table()
TN3 = c3[0,0]
TP3 = c3[1,1]
FN3 = c3[1,0]
FP3 = c3[0,1]
acc3 = (TN3 + TP3) / (TN3 + TP3 + FN3 + FP3)
print("accuracy: ", acc3)
```

```
Optimization terminated successfully.
         Current function value: 0.620588
         Iterations 5
Intercept   -0.628377
income      -0.101325
dtype: float64
accuracy:  0.6867
```

```
#vehicle_ownership
data['vehicle_ownership'] = data['vehicle_ownership'].astype('category').cat.codes
mdl4 = logit('outcome ~ vehicle_ownership', data = data).fit()
print(mdl4.params)
exp4 = pd.DataFrame({'vehicle_ownership':np.arange(0, 1)})
pre4 = exp4.assign(outcome = mdl4.predict(exp4))
c4 = mdl4.pred_table()
TN4 = c4[0,0]
TP4 = c4[1,1]
FN4 = c4[1,0]
FP4 = c4[0,1]
acc4 = (TN4 + TP4) / (TN4 + TP4 + FN4 + FP4)
print("accuracy: ", acc4)
```

```
Optimization terminated successfully.
         Current function value: 0.552412
         Iterations 5
Intercept            0.322231
vehicle_ownership   -1.724745
dtype: float64
accuracy:  0.7351
```

```
#vehicle_year
data['vehicle_year'] = data['vehicle_year'].astype('category').cat.codes
mdl5 = logit('outcome ~ vehicle_year', data = data).fit()
print(mdl5.params)
exp5 = pd.DataFrame({'vehicle_year':np.arange(0, 1)})
pre5 = exp5.assign(outcome = mdl5.predict(exp5))
c5 = mdl5.pred_table()
TN5 = c5[0,0]
TP5 = c5[1,1]
FN5 = c5[1,0]
FP5 = c5[0,1]
acc5 = (TN5 + TP5) / (TN5 + TP5 + FN5 + FP5)
print("accuracy: ", acc5)
```

```
Optimization terminated successfully.
         Current function value: 0.572668
         Iterations 6
Intercept      -2.127052
vehicle_year    1.735442
dtype: float64
accuracy:  0.6867
```

```
#credit score
data['credit_score'] = data['credit_score']
mdl6 = logit('outcome ~ credit_score', data = data).fit()
print(mdl6.params)
exp6 = pd.DataFrame({'vehicle_year':np.arange(0, 1)})
pre6 = exp6.assign(outcome = mdl6.predict(exp5))
c5 = mdl6.pred_table()
```

```
Optimization terminated successfully.
         Current function value: 0.567469
         Iterations 6
Intercept       1.949519
credit_score   -5.479372
dtype: float64
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
File /usr/local/lib/python3.8/dist-packages/patsy/compat.py:36, in call_and_wrap_exc(msg, origin, f, *args, **kwargs)
     35 try:
---> 36     return f(*args, **kwargs)
     37 except Exception as e:

File /usr/local/lib/python3.8/dist-packages/patsy/eval.py:165, in EvalEnvironment.eval(self, expr, source_name,
inner_namespace)
    164 code = compile(expr, source_name, "eval", self.flags, False)
--> 165 return eval(code, {}, VarLookupDict([inner_namespace]
    166                                     + self._namespaces))

File <string>:1

NameError: name 'credit_score' is not defined

The above exception was the direct cause of the following exception:

PatsyError                                Traceback (most recent call last)
File /usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:1104, in Results._transform_predict_exog(self, exog,
transform)
   1103 try:
-> 1104     exog = dmatrix(design_info, exog, return_type="dataframe")
   1105 except Exception as exc:

File /usr/local/lib/python3.8/dist-packages/patsy/highlevel.py:290, in dmatrix(formula_like, data, eval_env, NA_action,
return_type)
    289 eval_env = EvalEnvironment.capture(eval_env, reference=1)
--> 290 (lhs, rhs) = _do_highlevel_design(formula_like, data, eval_env,
    291                                   NA_action, return_type)
    292 if lhs.shape[1] != 0:

File /usr/local/lib/python3.8/dist-packages/patsy/highlevel.py:167, in _do_highlevel_design(formula_like, data, eval_env,
NA_action, return_type)
    166 if design_infos is not None:
--> 167     return build_design_matrices(design_infos, data,
    168                                  NA_action=NA_action,
    169                                  return_type=return_type)
    170 else:
    171     # No builders, but maybe we can still get matrices

File /usr/local/lib/python3.8/dist-packages/patsy/build.py:888, in build_design_matrices(design_infos, data, NA_action,
return_type, dtype)
    887 if factor_info not in factor_info_to_values:
--> 888     value, is_NA = _eval_factor(factor_info, data, NA_action)
    889     factor_info_to_isNAs[factor_info] = is_NA

File /usr/local/lib/python3.8/dist-packages/patsy/build.py:63, in _eval_factor(factor_info, data, NA_action)
     62 factor = factor_info.factor
---> 63 result = factor.eval(factor_info.state, data)
     64 # Returns either a 2d ndarray, or a DataFrame, plus is_NA mask
```

```
File /usr/local/lib/python3.8/dist-packages/patsy/eval.py:547, in EvalFactor._eval(self, code, memorize_state, data)
    546 inner_namespace = VarLookupDict([data, memorize_state["transforms"]])
--> 547 return call_and_wrap_exc("Error evaluating factor",
    548                          self,
    549                          memorize_state["eval_env"].eval,
    550                          code,
    551                          inner_namespace=inner_namespace)

File /usr/local/lib/python3.8/dist-packages/patsy/compat.py:43, in call_and_wrap_exc(msg, origin, f, *args, **kwargs)
     42     # Use 'exec' to hide this syntax from the Python 2 parser:
---> 43     exec("raise new_exc from e")
     44 else:
     45     # In python 2, we just let the original exception escape -- better
     46     # than destroying the traceback. But if it's a PatsyError, we can
     47     # at least set the origin properly.

File <string>:1

PatsyError: Error evaluating factor: NameError: name 'credit_score' is not defined
    outcome ~ credit_score
              ^^^^^^^^^^^^

During handling of the above exception, another exception occurred:

PatsyError                                Traceback (most recent call last)
Cell In[8], line 6
      4 print(mdl6.params)
      5 exp6 = pd.DataFrame({'vehicle_year':np.arange(0, 1)})
----> 6 pre6 = exp6.assign(outcome = mdl6.predict(exp5))
      7 c5 = mdl6.pred_table()

File /usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:1173, in Results.predict(self, exog, transform, *args,
**kwargs)
   1128 def predict(self, exog=None, transform=True, *args, **kwargs):
   1129     """
   1130     Call self.model.predict with self.params as the first argument.
   1131 
   (...)
   1171     returned prediction.
   1172     """
-> 1173     exog, exog_index = self._transform_predict_exog(exog,
   1174                                                     transform=transform)
   1176     predict_results = self.model.predict(self.params, exog, *args,
   1177                                          **kwargs)
   1179     if exog_index is not None and not hasattr(predict_results,
   1180                                               'predicted_values'):

File /usr/local/lib/python3.8/dist-packages/statsmodels/base/model.py:1111, in Results._transform_predict_exog(self, exog,
transform)
   1105 except Exception as exc:
   1106     msg = ('predict requires that you use a DataFrame when '
   1107            'predicting from a model\nthat was created using the '
   1108            'formula api.'
   1109            '\n\nThe original error message returned by patsy is:\n'
   1110            '{0}'.format(str(str(exc))))
-> 1111     raise exc.__class__(msg)
   1112 if orig_exog_len > len(exog) and not is_dict:
   1113     if exog_index is None:

PatsyError: predict requires that you use a DataFrame when predicting from a model
that was created using the formula api.

The original error message returned by patsy is:
Error evaluating factor: NameError: name 'credit_score' is not defined
    outcome ~ credit_score
              ^^^^^^^^^^^^
```