Modeling

April 22, 2021

[3]:
```
## Import packages
import os
import datetime
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pickle
import seaborn as sns
import shap
import xgboost as xgb
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold

# show plots in jupyter notebook
%matplotlib inline
# set plot style
sns.set(color_codes = True)
# set minimum number of columns to be displayed
pd.set_option('display.max_columns', 100)
# load JS visualization code to nodebook
shap.initjs()
```

```
<IPython.core.display.HTML object>
```

[8]:
```
## Data directory
DATA_DIR = os.path.join('..', 'BCG', 'processed_data')
TRAINING_DATA = os.path.join(DATA_DIR, 'train_data.pkl')
HISTORY_DATA = os.path.join(DATA_DIR, 'history_data.pkl')
```

[9]:
```
## Load data into a dataframe
train_data = pd.read_pickle(TRAINING_DATA)
history_data = pd.read_pickle(HISTORY_DATA)
## Merge data
train = pd.merge(train_data, history_data, on = 'id')
```

[10]:
```
train.columns
```

[10]:
```
Index(['id', 'cons_12m', 'cons_gas_12m', 'cons_last_month',
       'forecast_cons_12m', 'forecast_discount_energy',
       'forecast_meter_rent_12m', 'forecast_price_energy_p1',
       'forecast_price_energy_p2', 'forecast_price_pow_p1', 'has_gas',
       'imp_cons', 'margin_gross_pow_ele', 'margin_net_pow_ele', 'nb_prod_act',
       'net_margin', 'pow_max', 'churn', 'tenure', 'months_activ',
       'months_to_end', 'months_modif_prod', 'months_renewal', 'channel_epu',
       'channel_ewp', 'channel_fix', 'channel_foo', 'channel_lmk',
       'channel_sdd', 'channel_usi', 'origin_ewx', 'origin_kam', 'origin_ldk',
       'origin_lxi', 'origin_usa', 'activity_apd', 'activity_ckf',
       'activity_clu', 'activity_cwo', 'activity_fmw', 'activity_kkk',
       'activity_kwu', 'activity_sfi', 'activity_wxe',
       'mean_year_price_p1_var', 'mean_year_price_p2_var',
       'mean_year_price_p3_var', 'mean_year_price_p1_fix',
       'mean_year_price_p2_fix', 'mean_year_price_p3_fix',
       'mean_year_price_p1', 'mean_year_price_p2', 'mean_year_price_p3'],
      dtype='object')
```

1 Churn prediction with XGboost

Splitting Data First of all we will split the data into the variable that we are trying to predict y (churn) and those variables that we will use to predict churn X (the rest)

[11]:
```
y = train['churn']
X = train.drop(labels = ['id', 'churn'], axis = 1)
```

Next we will split the data into training and validation data. The percentages of each test can be changed but a 75% - 25% is a good ratio. We also use a random state generator in order to split it randomly.

[13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
↪random_state = 18)
```

Modelling

[15]:
```
model = xgb.XGBClassifier(learning_rate = 0.1,
                          max_depth = 6,
                          n_estimators = 500,
                          n_jobs = -1)
result = model.fit(X_train, y_train)
```

```
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].

[13:06:24] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

Modeling Evaluation ### Accuracy, Precision, Recall We are going to evaluate our Logistic Regression model on out test data (which we did not use for training) using the evaluation metrics of: > Accuracy: The most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations

Precision: The ratio of correctly predicted positive observations to the total predicted positive observations

Recall (Sensitivity): The ratio of correctly predicted positive observations to the all observations in actual class

[14]:
```
def evaluate(model_, X_test_, y_test_):
    '''
    Evaluate the accuracy, precision and recall of a model
    '''
    # Get the model predictions
    prediction_test_ = model_.predict(X_test_)
    # Print the evaluation metrics as pandas dataframe
    results = pd.DataFrame({'Accuracy': [metrics.accuracy_score(y_test_,␣
↪prediction_test_)],
                            'Precision': [metrics.precision_score(y_test_,␣
↪prediction_test_)],
                            'Recall': [metrics.recall_score(y_test_,␣
↪prediction_test_)]})
    # For a more detailed report
    # print(metrics.classification_report(y_test_, prediction_test_))
    return results
```

[16]:
```
evaluate(model, X_test, y_test)
```

[16]:

	Accuracy	Precision	Recall
0	0.906809	0.738095	0.149398

ROC-AUC Receiver Operating Characteristic (ROC) curve is a plot of the true positive rate against false positive rate. It shows the tradeoff between sensitivity and specificity.

In a nutshell, it tells how much model is capable of distinguishing between classes.

[23]:

```
def calculate_roc_auc(model_, X_test_, y_test_):
    '''
    Evaluate the roc-auc score
    '''
    # Get the model predictions
    # Note that we using the prediction for the class 1 -> churn
    prediction_test_ = model_.predict_proba(X_test_)[:, 1]
    # compute roc-auc
    fpr, tpr, thresholds = metrics.roc_curve(y_test_, prediction_test_)
    # print the evaluation metrics as opandas dataframe
    score = pd.DataFrame({'ROC-AUC': [metrics.auc(fpr, tpr)]})
    return fpr, tpr, score

def plot_roc_auc(fpr, tpr):
    '''
    Plotmthe Receiver Operating Characteristic from a list
    of true positive rates and false positive rates
    '''
    # Initialize plot
    f, ax = plt.subplots(figsize = (14, 8))
    # Plot ROC
    roc_auc = metrics.auc(fpr, tpr)
    ax.plot(fpr, tpr,
            lw = 2,
            alpha = 0.3,
            label = 'AUC = %0.2f' % (roc_auc)
           )
    # Plot the random line
    plt.plot([0,1], [0, 1],
             linestyle = '--',
            lw = 3,
            color = 'r',
            label = 'Random',
            alpha = 0.8)
    # Fine tune and show the plot
    ax.set_xlim([-0.05, 1.05])
    ax.set_ylim([-0.05, 1.05])
    ax.set_xlabel('False Positive Rate (FPR)')
    ax.set_ylabel('True Positive Rate (TPR)')
    ax.set_title('ROC-AUC')
```

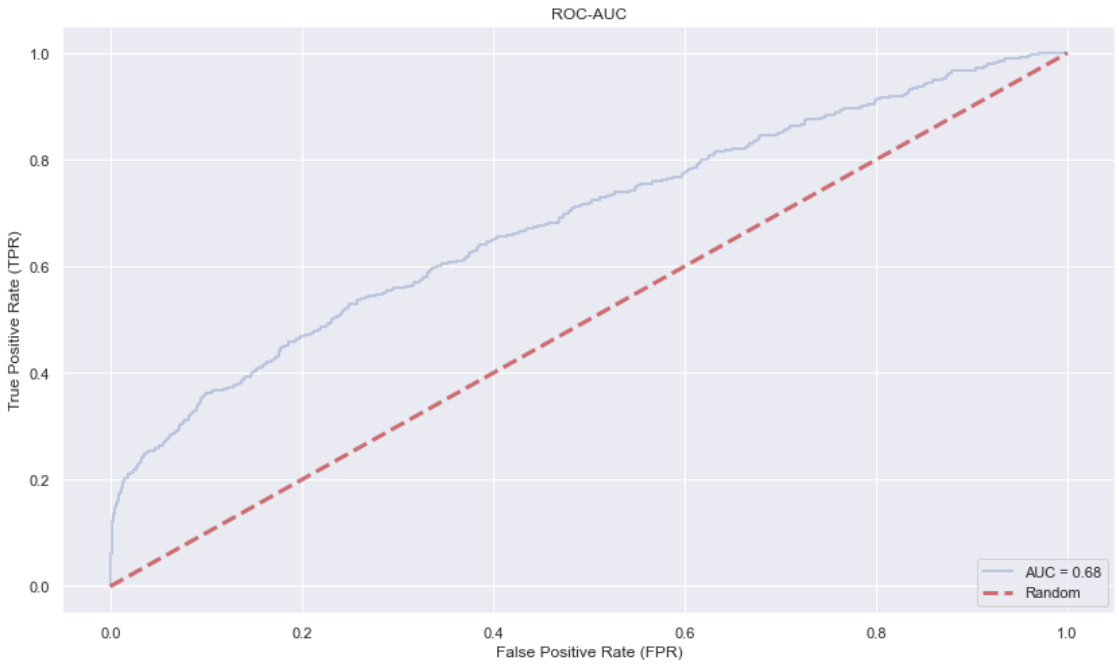
```
    ax.legend(loc = 'lower right')
    plt.show()
```

[24]:
```
fpr, tpr, auc_score = calculate_roc_auc(model, X_test, y_test)
```

[25]:
```
auc_score
```

[25]:
```
    ROC-AUC
0  0.684365
```

[26]:
```
plot_roc_auc(fpr, tpr)
plt.show()
```



Stratified K-fold validation

[41]:
```
def plot_roc_curve(fprs, tprs):
    '''
    Plot the Receiver OOperating Characteristic from a list of true positive␣
 ↪rates
    and false positve rates.
    '''
    # Initialize useful lists + the plots axes
    tprs_interp = []
    aucs = []
    mean_fpr = np.linspace(0, 1, 100)
    f, ax = plt.subplots(figsize = (18, 10))
```

```
    # Plot ROC for each k-fold + comute AUC scores
    for i, (fpr, tpr) in enumerate(zip(fprs, tprs)):
        tprs_interp.append(np.interp(mean_fpr, fpr, tpr))
        tprs_interp[-1][0] = 0.0
        roc_auc = metrics.auc(fpr, tpr)
        aucs.append(roc_auc)
        ax.plot(fpr, tpr,
                lw = 2,
                alpha = 0.3,
                label = 'ROC fold %d (AUC = %0.2F)'%(i, roc_auc))
    # Plot the luck line
    plt.plot([0,1], [0,1],
            linestyle = '--',
            lw = 3,
            color = 'r',
            label = 'Random',
            alpha = 0.8)
    # Plot the mean ROC
    mean_tpr = np.mean(tprs_interp, axis = 0)
    mean_tpr[-1] = 1.0
    mean_auc = metrics.auc(mean_fpr, mean_tpr)
    std_auc = np.std(aucs)
    ax.plot(mean_fpr, mean_tpr,
            color = 'b',
            label = r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
            lw = 4,
           alpha = 0.8)
    # Plot the Standard deviation around the mean ROC
    std_tpr = np.std(tprs_interp, axis = 0)
    tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
    tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
    ax.fill_between(mean_fpr, tprs_lower, tprs_upper,
                    color = 'grey',
                    alpha = 0.2,
                    label = r'$\pm$ 1 std. dev.')

    # Fine tune and show the plot
    ax.set_xlim([-0.05, 1.05])
    ax.set_ylim([-0.05, 1.05])
    ax.set_xlabel('False Positive Rate (FPR)')
    ax.set_ylabel('True Positive Rate (TPR)')
    ax.set_title('ROC-AUC')
    ax.legend(loc = 'lower right')
    plt.show()
    return (f, ax)
```

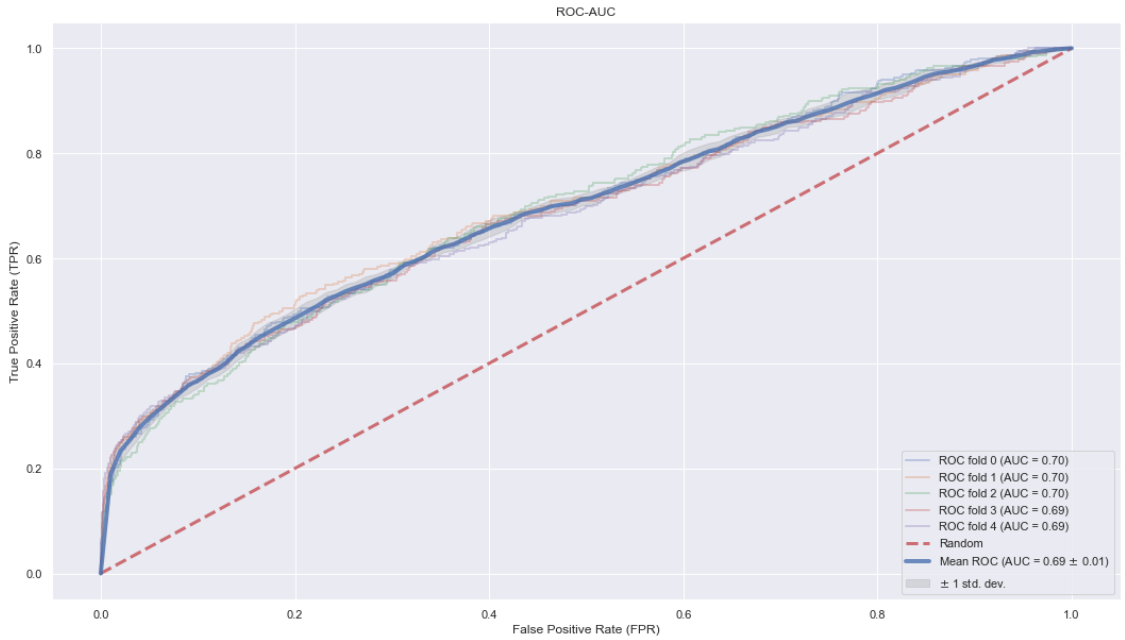
```
def compute_roc_auc(model_, index):
    y_predict = model_.predict_proba(X.iloc[index])[:, 1]
    fpr, tpr, thresholds = metrics.roc_curve(y.iloc[index], y_predict)
    auc_score = metrics.auc(fpr, tpr)
    return fpr, tpr, auc_score
```

[33]:
```
cv = StratifiedKFold(n_splits = 5, random_state = 13, shuffle = True)
fprs, tprs, scores = [], [], []
```

[35]:
```
for (train, test), i in zip(cv.split(X,y), range(5)):
    model.fit(X.iloc[train], y.iloc[train])
    _, _, auc_score_train = compute_roc_auc(model, train)
    fpr, tpr, auc_score = compute_roc_auc(model, test)
    scores.append((auc_score_train, auc_score))
    fprs.append(fpr)
    tprs.append(tpr)
```

```
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].

[14:38:16] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[14:38:22] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[14:38:28] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[14:38:35] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[14:38:42] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

[42]:
```
plot_roc_curve(fprs, tprs)
plt.show()
```



Model Finetuning ### Random Search Cross Validation

[44]:
```
from sklearn.model_selection import RandomizedSearchCV
```

[45]:
```
# Create the random grid
params = {
    'min_child_weight': [i for i in np.arange(1, 15, 1)],
    'gamma': [i for i in np.arange(0, 6, 0.5)],
    'subsample': [i for i in np.arange(0, 1.1, 0.1)],
    'colsample_bytree': [i for i in np.arange(0, 1.1, 0.1)],
    'max_depth': [i for i in np.arange(1, 15, 1)],
    'scale_pos_weight': [i for i in np.arange(1, 15, 1)],
    'learning_rate': [i for i in np.arange(0, 0.15, 0.01)],
    'n_estimators': [i for i in np.arange(0, 2000, 100)]

}
```

[48]:
```
# create base model
xg = xgb.XGBClassifier(objective = 'binary:logistic',
                       silent = True,
                       nthread = 1)
```

[49]:
```
# Random search of parameters, using 5
xg_random = RandomizedSearchCV(xg, param_distributions = params,
                               n_iter = 1,
                               scoring = 'roc_auc',
```

```
                                n_jobs = 4,
                                cv = 5,
                                verbose = 3,
                                random_state = 1001)
# Fit the random search model
xg_random.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   2 out of   5 | elapsed:    0.9s remaining:    1.4s
[Parallel(n_jobs=4)]: Done   5 out of   5 | elapsed:    1.8s finished
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].

[14:55:40] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[14:55:40] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

[49]:
```
RandomizedSearchCV(cv=5,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, gamma=None,
                                           gpu_id=None, importance_type='gain',
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=None,
                                           min_child_weight=None, missing=nan,
                                           monotone_constraints=None,
                                           n_estimators=100,…
                                'min_child_weight': [1, 2, 3, 4, 5, 6,
                                                     7, 8, 9, 10, 11,
                                                     12, 13, 14],
                                'n_estimators': [0, 100, 200, 300, 400,
```

```
                                                    500, 600, 700, 800,
                                                    900, 1000, 1100, 1200,
                                                    1300, 1400, 1500, 1600,
                                                    1700, 1800, 1900],
                                        'scale_pos_weight': [1, 2, 3, 4, 5, 6,
                                                             7, 8, 9, 10, 11,
                                                             12, 13, 14],
                                        'subsample': [0.0, 0.1, 0.2,
                                                      0.30000000000000004, 0.4,
                                                      0.5, 0.6000000000000001,
                                                      0.7000000000000001, 0.8,
                                                      0.9, 1.0]},
                            random_state=1001, scoring='roc_auc', verbose=3)
```

[70]:
```
best_random = xg_random.best_params_
print(best_random)
```

```
{'subsample': 0.0, 'scale_pos_weight': 6, 'n_estimators': 300,
'min_child_weight': 10, 'max_depth': 12, 'learning_rate': 0.1, 'gamma': 2.5,
'colsample_bytree': 1.0}
```

[75]:
```
best_random = {'subsample': 0.8,
               'scale_pos_weight': 1,
               'n_estimators': 1100,
               'min_child_weight': 1,
               'max_depth': 12,
               'learning_rate': 0.01,
               'gamma': 4.0,
               'colsample_bytree': 0.60}
```

[76]:
```
# create a model with the parameters found
model_random = xgb.XGBClassifier(objective = 'binary:logistic',
                                 silent = True,
                                 nthread = 1,
                                 **best_random)
fprs, tprs, scores = [],[],[]
```

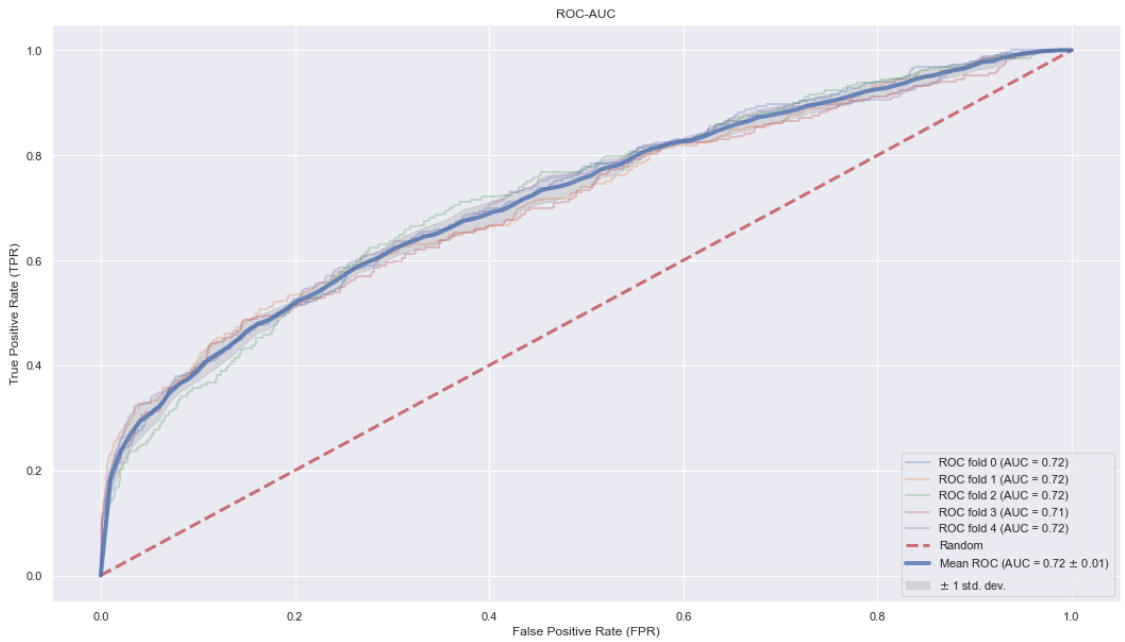
[77]:
```
for (train, test), i in zip(cv.split(X,y), range(5)):
    model_random.fit(X.iloc[train], y.iloc[train])
    _, _, auc_score_train = compute_roc_auc(model_random, train)
    fpr, tpr, auc_score = compute_roc_auc(model_random, test)
    scores.append((auc_score_train, auc_score))
    fprs.append(fpr)
    tprs.append(tpr)
```

```
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
```

```
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].

[16:02:45] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:02:45] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[16:03:47] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:03:48] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[16:04:50] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:04:50] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[16:05:53] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.
```

```
  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:05:53] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[16:06:56] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:06:56] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

[78]:
```
plot_roc_curve(fprs, tprs)
plt.show()
```



1.0.1 Grid Search with Cross Validation

[80]:
```
from sklearn.model_selection import GridSearchCV
```

[81]:
```
# Create the parameter grid based on the results of random search
param_grid = {
    'subsample': [0.7],
    'scale_pos_weight': [1],
    'n_estimators': [1100],
    'min_child_weight': [1],
    'max_depth': [12, 13, 14],
    'learning_rate': [0.005, 0.01],
    'gamma': [4.0],
    'colsample_bytree': [0.6]
}
```

[82]:
```
# create model
xg = xgb.XGBClassifier(objective = 'binary:logistic',
                       silent = True,
                       nthread = 1)
```

[83]:
```
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = xg,
                           param_grid = param_grid,
                           cv = 5,
                           n_jobs = -1,
                           verbose = 2,
                           scoring = 'roc_auc')
```

[84]:
```
# Fit the grid search to the data
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  23 out of  30 | elapsed:  5.4min remaining:  1.7min
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:  7.2min finished
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].

[16:16:32] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
```

```
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:16:32] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

[84]:
```
GridSearchCV(cv=5,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, gamma=None,
                                     gpu_id=None, importance_type='gain',
                                     interaction_constraints=None,
                                     learning_rate=None, max_delta_step=None,
                                     max_depth=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     n_estimators=100, n_jobs…
                                     reg_alpha=None, reg_lambda=None,
                                     scale_pos_weight=None, silent=True,
                                     subsample=None, tree_method=None,
                                     validate_parameters=None, verbosity=None),
             n_jobs=-1,
             param_grid={'colsample_bytree': [0.6], 'gamma': [4.0],
                         'learning_rate': [0.005, 0.01],
                         'max_depth': [12, 13, 14], 'min_child_weight': [1],
                         'n_estimators': [1100], 'scale_pos_weight': [1],
                         'subsample': [0.7]},
             scoring='roc_auc', verbose=2)
```

[85]:
```
best_grid = grid_search.best_params_
best_grid
```

[85]:
```
{'colsample_bytree': 0.6,
 'gamma': 4.0,
 'learning_rate': 0.005,
 'max_depth': 12,
 'min_child_weight': 1,
 'n_estimators': 1100,
 'scale_pos_weight': 1,
 'subsample': 0.7}
```

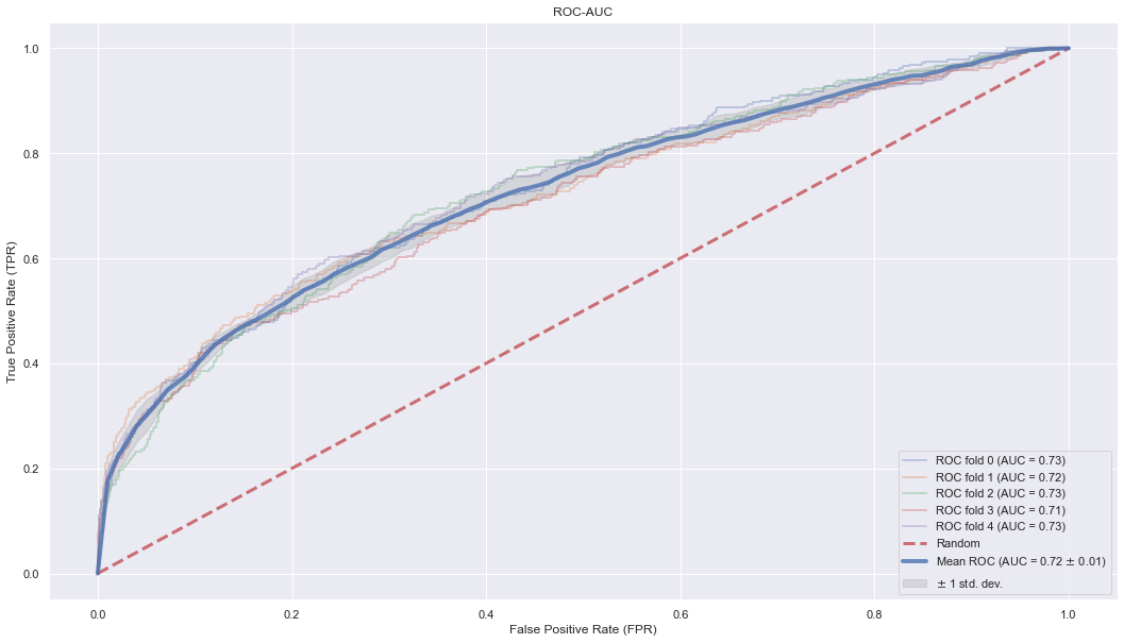
[88]:
```
# create a model with the parameters found
model_grid = xgb.XGBClassifier(objective = 'binary:logistic',
                               silent = True,
                               nthread = 1,
                               **best_grid)
fprs, tprs, scores = [], [], []
```

[89]:
```
for (train, test), i in zip(cv.split(X, y), range(5)):
    model_grid.fit(X.iloc[train], y.iloc[train])
    _, _, auc_score_train = compute_roc_auc(model_grid, train)
    fpr, tpr, auc_score = compute_roc_auc(model_grid, test)
    scores.append((auc_score_train, auc_score))
    fprs.append(fpr)
    tprs.append(tpr)
```

```
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].

[16:23:27] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:23:27] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[16:24:31] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:24:32] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

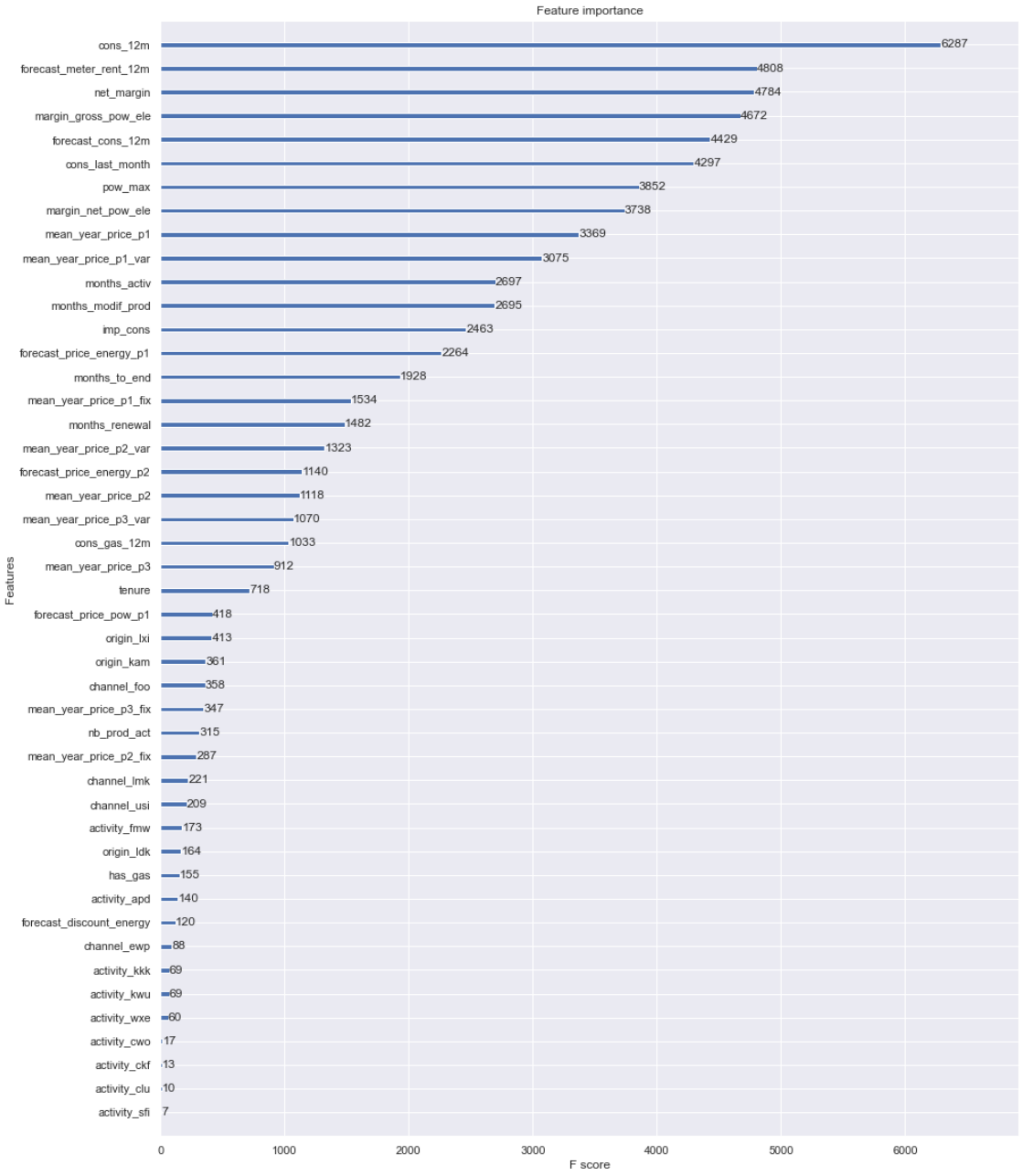
```
[16:25:35] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:25:35] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[16:26:39] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:26:39] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[16:27:42] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:27:42] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

[90]:
```
plot_roc_curve(fprs, tprs)
plt.show()
```



Understanding the model ### Feature Importance One simple way of observing the feature importance is through counting the number of times each feature is split on across all boosting rounds (trees) in the model, and then visualizing the result as a bar graph, with the features ordered according to how many times they appear.

[92]:
```
fig, ax = plt.subplots(figsize = (15, 20))
xgb.plot_importance(model_grid, ax=ax);
```



In the feature importance graph above we can see that `cons_12m` and `net_margin` are the features that appear the most in our model and we could infer that these two features have a significant importance in our model.

Partial Dependence Plot

[95]:
```
from sklearn.inspection import plot_partial_dependence
```

Because currently there is a bug that does not allow us to use our trained model with oandas dataframes, we will areplica and train it using numpy arrays

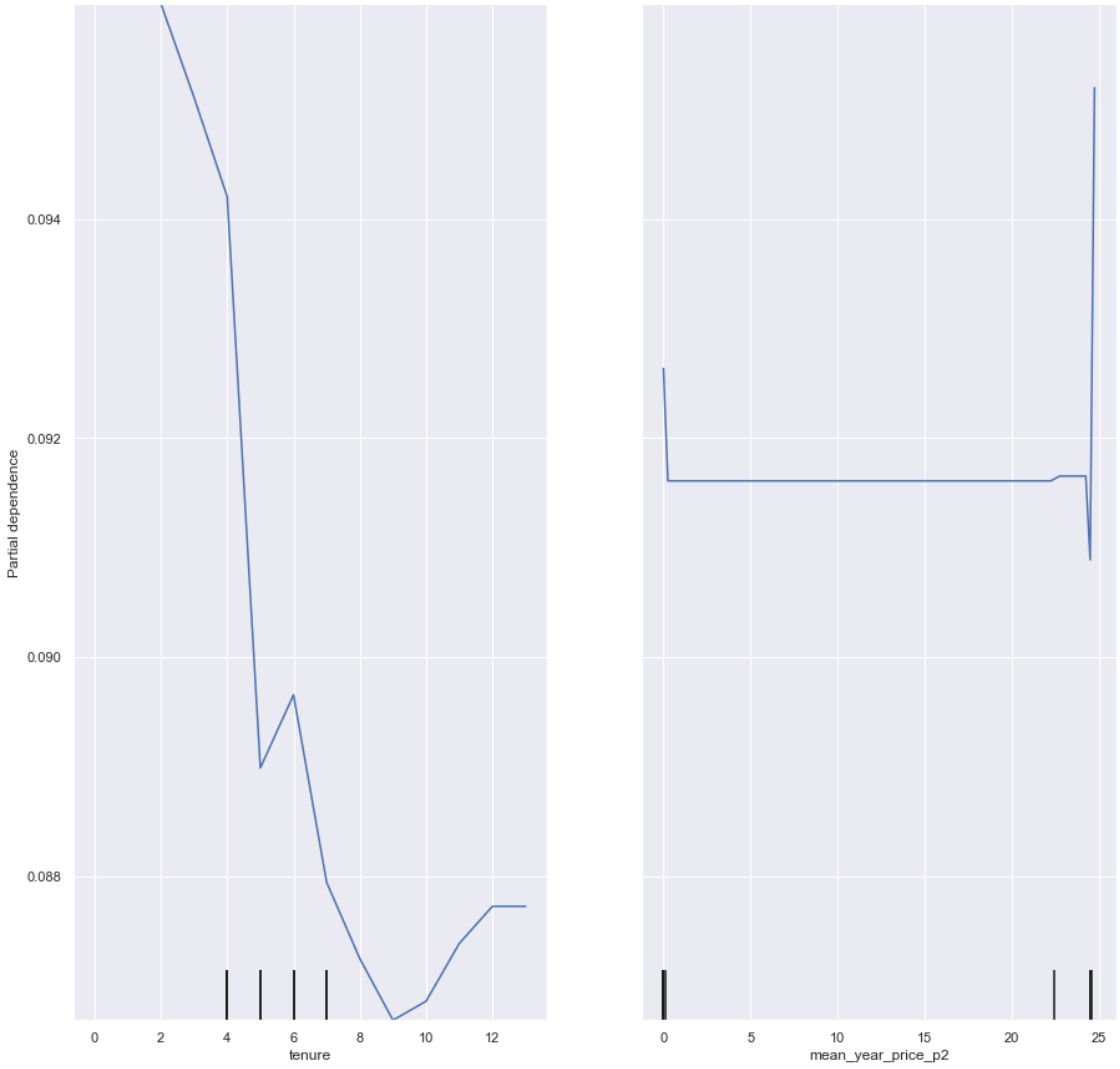
[99]:
```
# create a model with the parameters found
model_grid_v2 = xgb.XGBClassifier(objective = 'binary:logistic',
                                  silent = True,
                                  nthread = 1,
                                  **best_grid)
model_grid_v2.fit(X_train.values, y_train.values)
```

```
The use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].

[16:41:39] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:541:
Parameters: { silent } might not be used.

  This may not be accurate due to some parameters are only used in language
bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through
this
  verification. Please open an issue if you find above cases.


[16:41:39] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1061:
Starting in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
```

[99]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.6, gamma=4.0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.005, max_delta_step=0, max_depth=12,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=1100, n_jobs=1, nthread=1, num_parallel_tree=1,
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
              silent=True, subsample=0.7, tree_method='exact',
              validate_parameters=1, verbosity=None)
```

[111]:
```
fig = plt.figure(figsize = (15, 15))
plot_partial_dependence(model_grid_v2, X_test.values,
                        features = [16, 49],
                        feature_names = X_test.columns.tolist(),
                        fig = fig);
```

```
The fig parameter is deprecated in version 0.22 and will be removed in version
0.24
```



Comparing the PDP plots with respect to our previous models, we can see how they are slightly different. `tenure`

The overall trend is unchanged as compared to our previoous models, we cansee how they are slighly different times of the tenure (6y) but then it goes down again and bottoms around 10 years. Then, it starts recovering a bit.

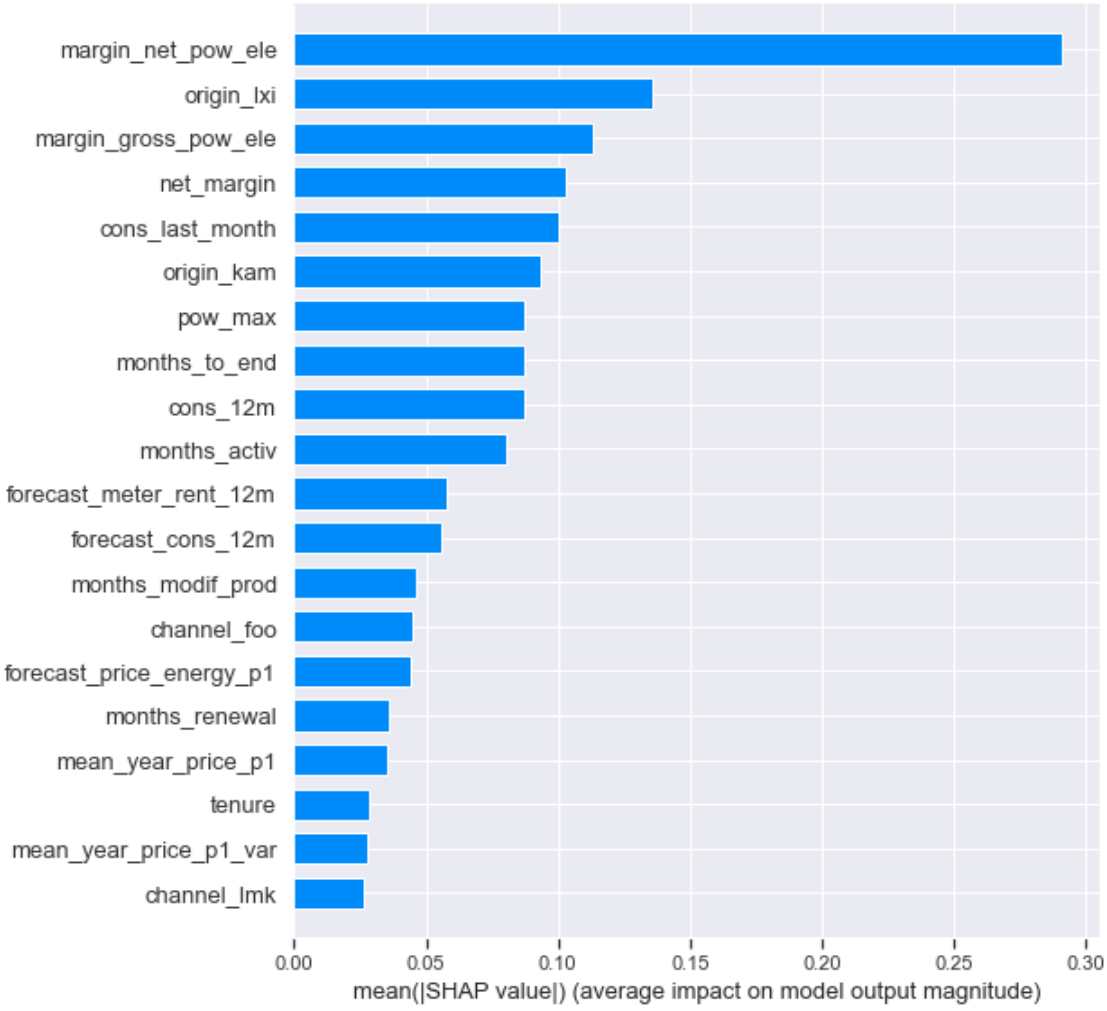
`mean_year_price_p2`

In our previous models, we saw a sort of 'stairshape', in this case we see the PDP is almost flat with some spikes on the extreme values, whcih hints us that the variable `mean_year_price_p2` is not very relevant in this model.

SHAP - Feature Importance

[112]:
```
explainer = shap.TreeExplainer(model_grid)
shap_values = explainer.shap_values(X_test)
```

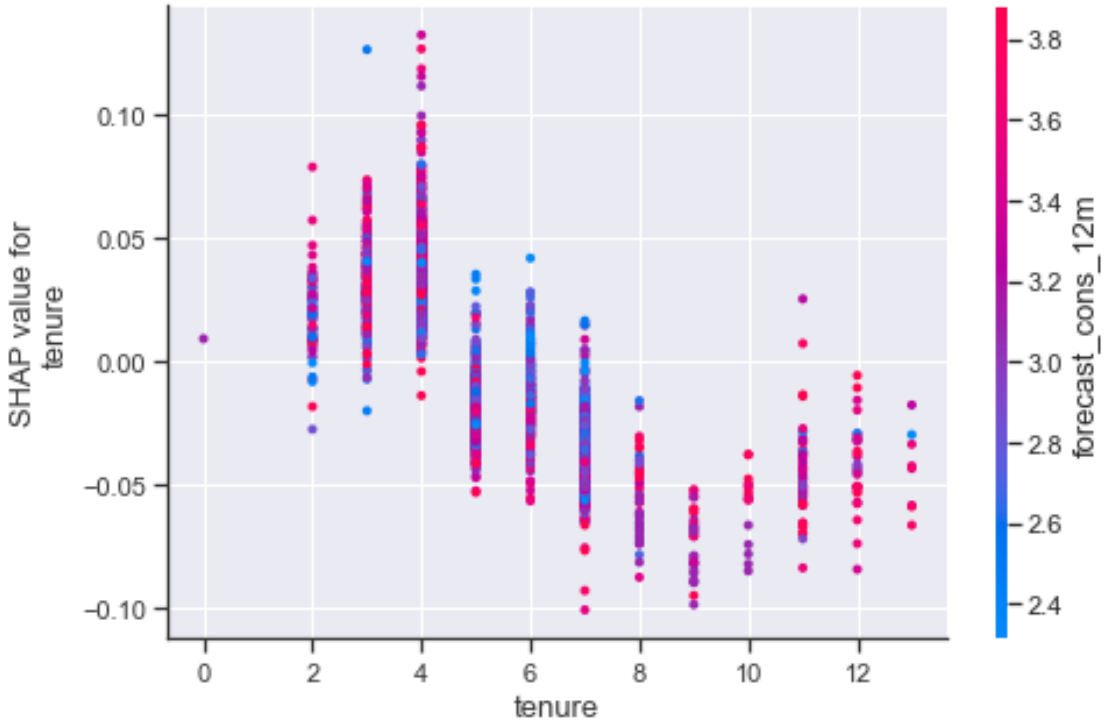
[113]:
```
# Feature importance for class 1 - churn
shap.summary_plot(shap_values,
                  X_test,
                  plot_type = 'bar')
```



As expeced the `margin_net_pow_ele` is most important feature by far. It is interesting to compare how much important the top feature becomes in contrast with the other models we created `Randon forest` and `Logistic Regression`

1.0.2 SHAP - Partial Dependence Plot

[114]:
```
shap.dependence_plot('tenure', shap_values, X_test) #, interaction_index =␣
↪'origin_lxi
```

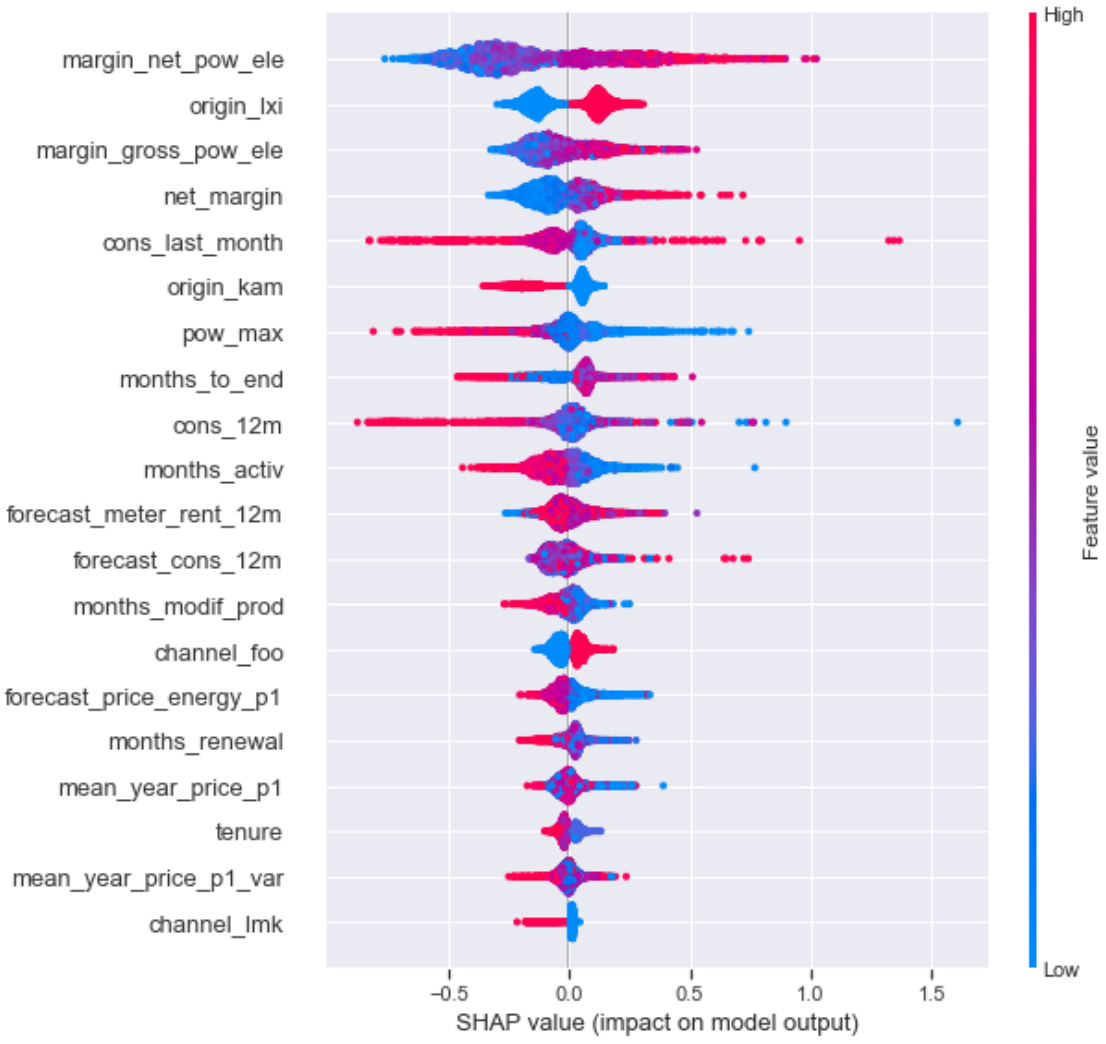


In this case we see a much clearer pattern, in which the longer the tenure the less likely the company is, sort of decreasing linearly until it bottoms around 9y of tenure. From year 10 of tenure, the churn increases again.

[115]:
```
shap.force_plot(explainer.expected_value,
                shap_values[4023],
                X_test.iloc[4023, :],
                link = 'logit')
```

[115]:
```
<shap.plots._force.AdditiveForceVisualizer at 0x7fc1b2fc4f90>
```

[119]:
```
shap.summary_plot(shap_values, X_test);
```


margin_net_pow_ele
origin_lxi
margin_gross_pow_ele
net_margin
cons_last_month
origin_kam
pow_max
months_to_end
cons_12m
months_activ
forecast_meter_rent_12m
forecast_cons_12m
months_modif_prod
channel_foo
forecast_price_energy_p1
months_renewal
mean_year_price_p1
tenure
mean_year_price_p1_var
channel_lmk
−0.5
0.0
0.5
1.0
1.5
SHAP value (impact on model output)
High
Feature value
Low

[ ]: