Deposit Subscription - Predictive Modeling -

BY: MAI HAN TRAN

AGENDA OF ITEMS

BUSINESS GOAL

DATA QUALITY ASSESSMENT

DATA EXPLORATION

METHODOLOGY

MODEL SELECTION

BUSINESS GOAL:

- A banking institution decided to pursue a direct marketing campaign to increase number of deposit subscriptions.
- A prediction model via Python was developed to help the bank identify most potential subscribers to enhance campaign success and best leverage its budget.

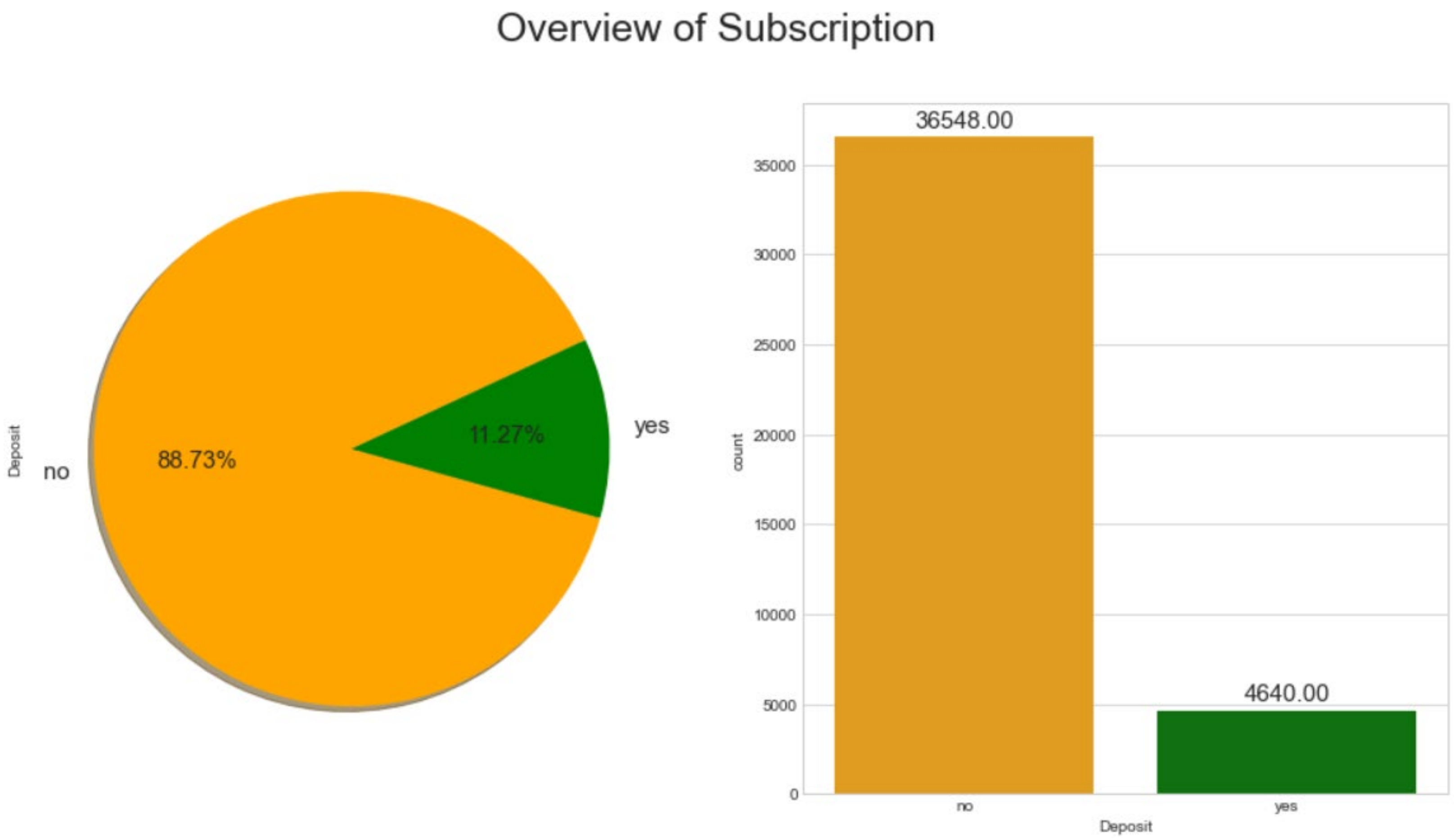
- *About the dataset:* The dataset was called "Bank Marketing Data", which was taken from UCI Machine Learning Repository

DATA QUALITY ASSESSMENT

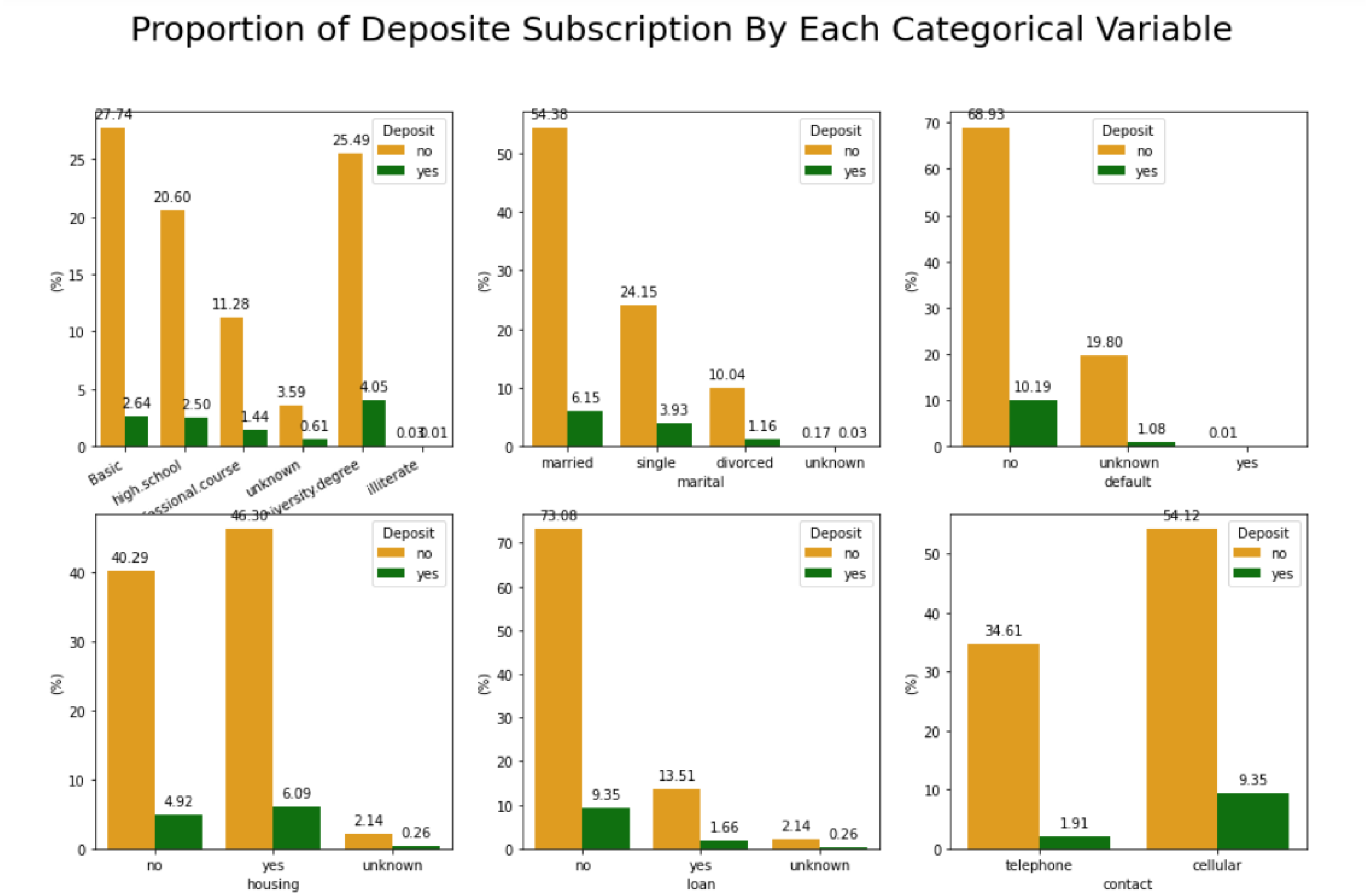
	OVERVIEW	MISSING VALUE	DATA INCONSISTENCY	DATA REGROUP
DATASET	41,188 records 21 data fields	NONE	NONE	Education field
ACTION	N/A	N/A	N/A	Regroup "*basic.4y*", "*basic.6y*", "*basic.9y*" into a new category called "basic" under education field

DATA EXPLORATION

- Number of people who did not have deposit subscription was prevalent within the dataset.
- The dominance of this population must be addressed in the analysis to improve prediction accuracy.



DATA EXPLORATION



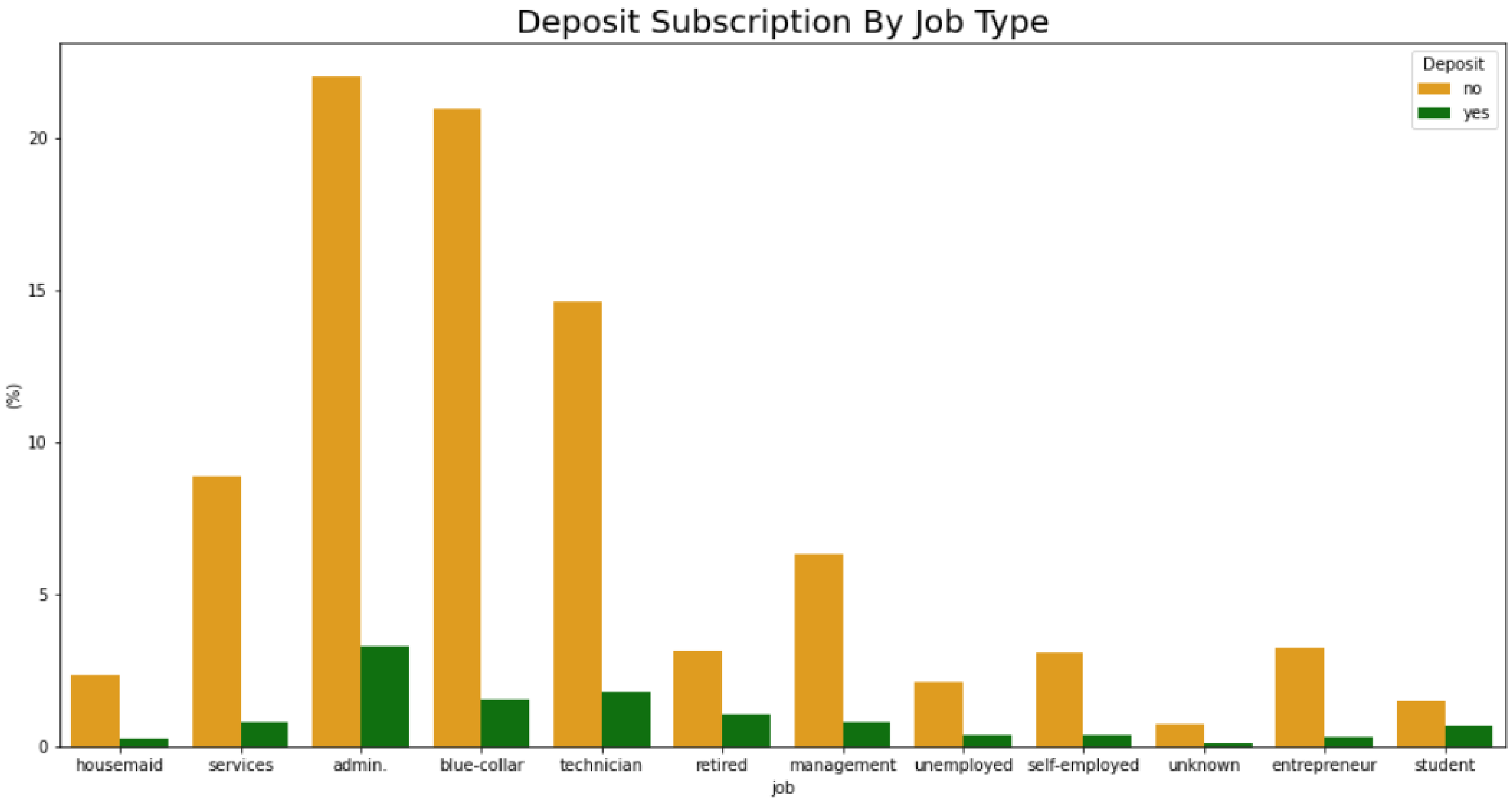
- ❑ Number of deposit subscription came from the majority of:

  - Individuals with university degree
  - Married individuals
  - Individuals without credit default
  - Individuals with a housing loan
  - Individuals without a personal loan
  - Individuals who had cellular

- ➢ As a result, these 6 categorical variables are good predictors for model development.

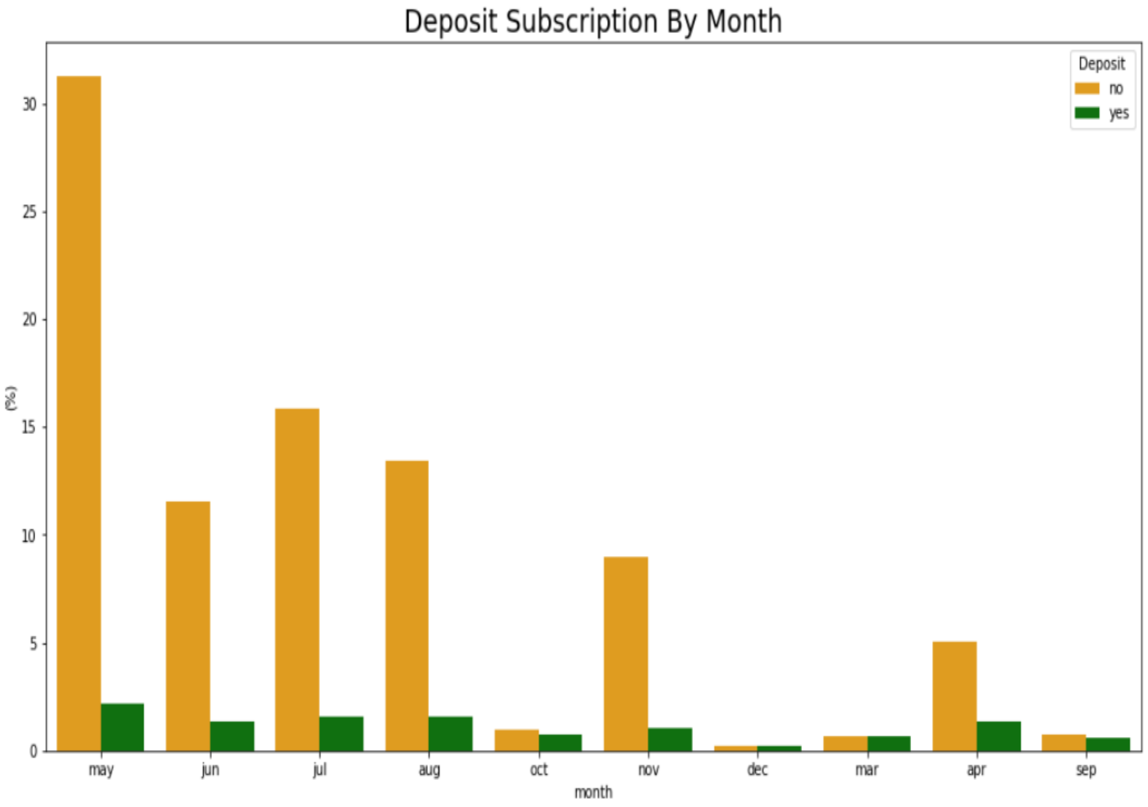
DATA EXPLORATION

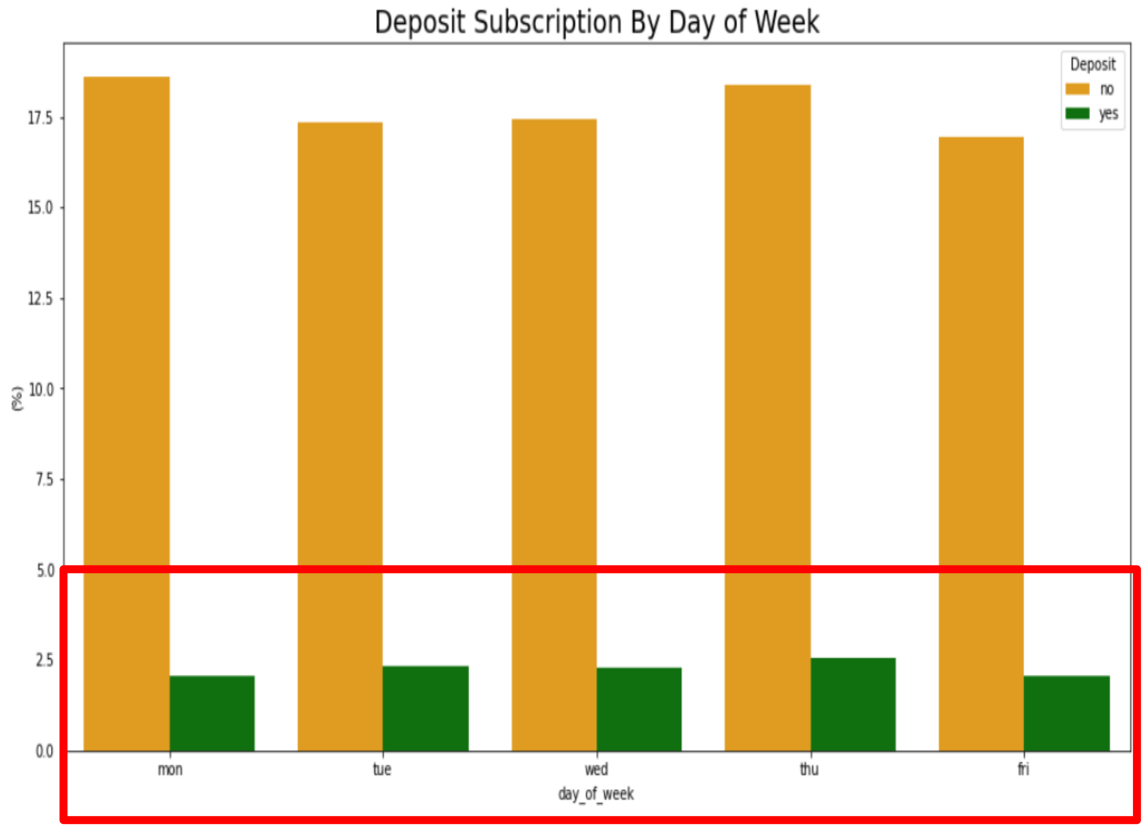
- Blue-collar workers, technicians, and administrators were mainly the deposit subscription holders.
- Job type should be included as a feature in the prediction model.



DATA EXPLORATION

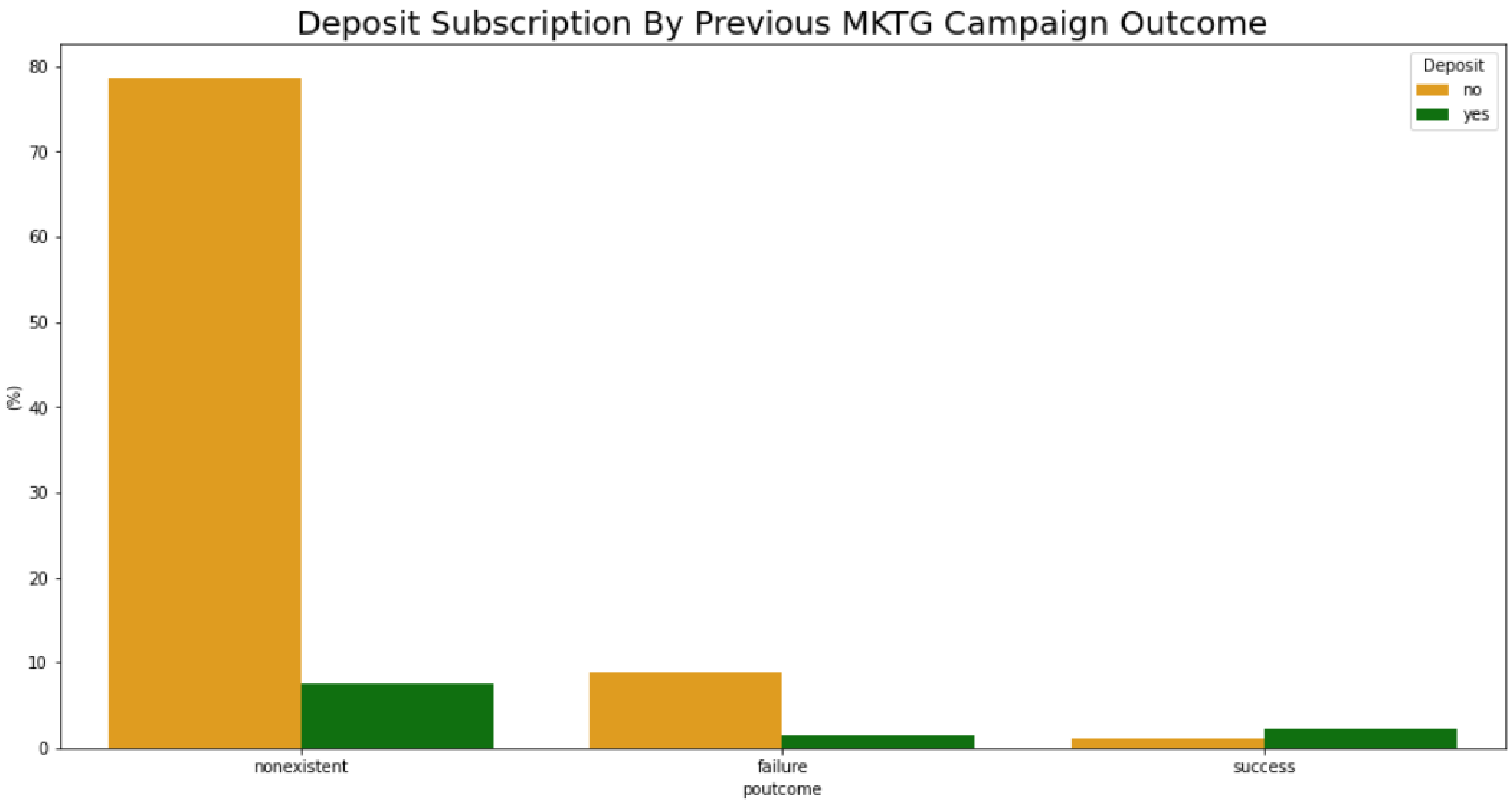
- Number of deposit subscriptions peaked in the month of April, May, June, July. However, number of deposit subscriptions were equally scattered across the day of week.
- Therefore, the prediction model would include *"month"* and exclude *"day_of_week"* variable.





DATA EXPLORATION

- Number of deposit subscriptions peaked for successful and non-existent result in previous marketing campaign. Therefore, *"poutcome" was inferred as good predictor for the model.*

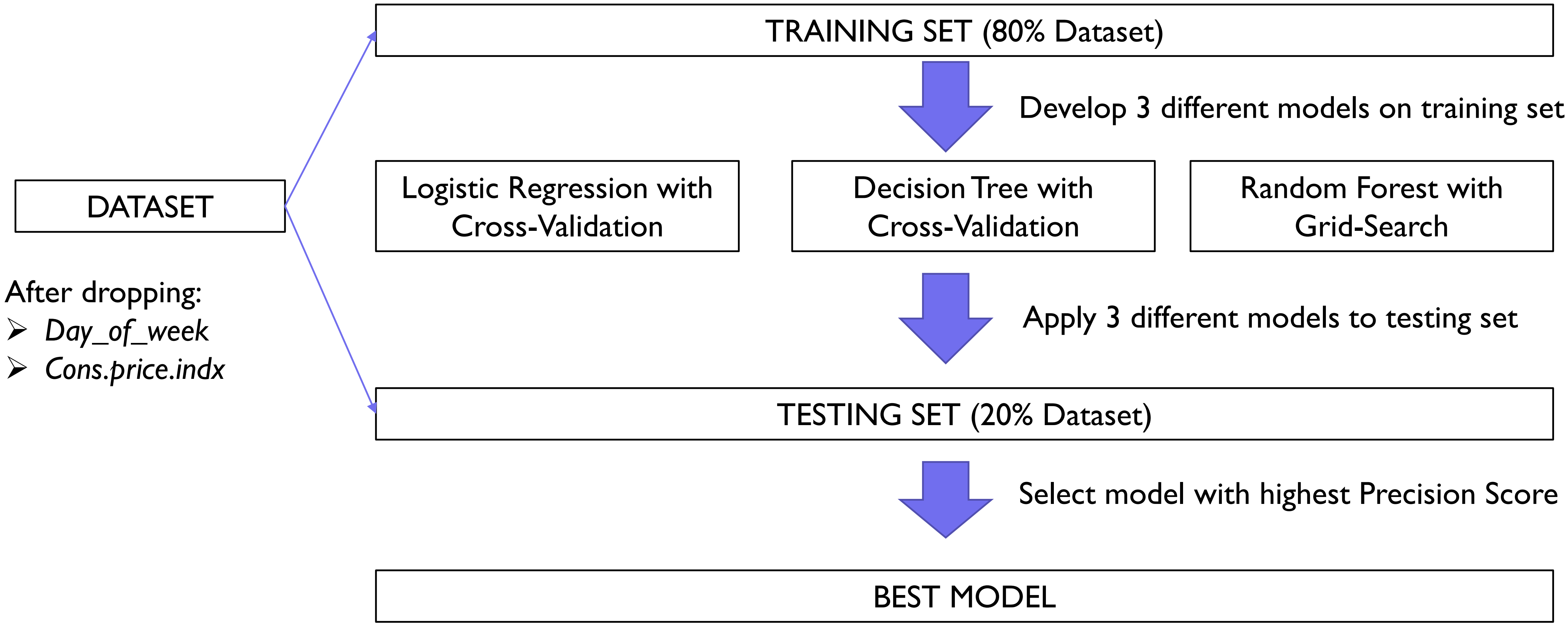


DATA EXPLORATION

- There was no significant difference between people who had deposit subscription vs. who did not.
- Therefore, this variable would be removed from the prediction model.

Deposit	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
no	39.91	220.84	2.63	984.11	0.13	0.25	93.60	-40.59	3.81	5176.17
yes	40.91	553.19	2.05	792.04	0.49	-1.23	93.35	-39.79	2.12	5095.12

METHODOLOGY



MODEL SELECTION

- Out of 3 models, Decision Tree has slightly higher precision score than the others.
- As a result, Decision Tree is concluded to be the best prediction model to measure result of future direct marketing campaign.

Logistic Regression with Cross-Validation

	precision	recall	f1-score	support
0	0.93	0.98	0.95	7319
1	0.66	0.37	0.48	919
accuracy			0.91	8238
macro avg	0.79	0.68	0.71	8238
weighted avg	0.90	0.91	0.90	8238

Decision Tree with Cross-Validation

	precision	recall	f1-score	support
0	0.94	0.96	0.95	7319
1	0.60	0.55	0.57	919
accuracy			0.91	8238
macro avg	0.77	0.75	0.76	8238
weighted avg	0.91	0.91	0.91	8238

Random Forest with Grid-Search

	precision	recall	f1-score	support
0	0.90	1.00	0.95	7319
1	0.80	0.16	0.26	919
accuracy			0.90	8238
macro avg	0.85	0.58	0.60	8238
weighted avg	0.89	0.90	0.87	8238

Note: Precision Score = % of predicted deposit subscriptions relative to all direct marketing efforts.
Higher "Precision Score" means higher ROI on ad spend

THANK YOU

APPENDIX – PYTHON CODE

```
### LOGISTIC REGRESSION WITH CROSS VALIDATION
k_fold = KFold(n_splits = 10, random_state = 0)
lgt = LogisticRegressionCV(cv=k_fold,scoring='precision')
lgt_y_pred = lgt.fit(x_train,y_train).predict(x_test)
lgt_precision = precision_score(y_test, lgt_y_pred,average='weighted')
print(round(lgt_precision,3))
print(classification_report(y_test, lgt_y_pred))
```

```
### DECISION TREE MODEL WITH CROSS VALIDATION
# Iteration over various tree depths to idenitfy the best precision score

for max_depth_val in [2,3,5,6,7,10,12]:
    k_fold = KFold(n_splits = 10, random_state = 0)
    clf = DecisionTreeClassifier(max_depth = max_depth_val)
    print("Evaluating Decision Tree for max_depth = %s" %(max_depth_val))
    y_pred = clf.fit(x_train, y_train).predict(x_test)

# Calculate precision for cross validation and test
    cv_precision = cross_val_score(
        clf, x_train, y_train, cv = k_fold, scoring = 'precision_weighted')
    precision = precision_score(y_test, y_pred, average = 'weighted')
    print("Cross validation Precision: %s" %(cv_precision))
    print("Test Precision: %s" %(round(precision,4)))

# Best Decision Tree with Max Dep = 6
k_fold = KFold(n_splits = 10, random_state = 0)
dt = DecisionTreeClassifier(max_depth = max_depth_val)
dt_y_pred = dt.fit(x_train,y_train).predict(x_test)
precision_dt = precision_score(y_test,dt_y_pred, average = 'weighted')
print(precision_dt)
print(classification_report(y_test, dt_y_pred))
```

```
### RANDOM FOREST MODEL WITH HYPERPARAMETER TUNING
# Create list of hyperparameters
n_estimators = [2, 80]
max_depth = [2, 100]
param_grid = {'n_estimators': n_estimators, 'max_depth': max_depth}

# Use Grid search CV to find best parameters
print("starting RF grid search.. ")
k_fold = KFold(n_splits = 10, random_state = 0)
rf = RandomForestClassifier()
rf_grid = GridSearchCV(estimator = rf, param_grid = param_grid, scoring = 'precision',cv=k_fold)
rf_y_pred = rf_grid.fit(x_train, y_train).predict(x_test)
rf_precision = precision_score(y_test, rf_y_pred, average="weighted")
print(rf_precision)
print(classification_report(y_test, rf_y_pred))
```