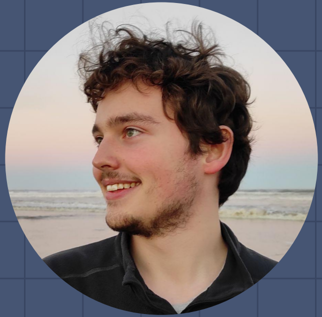
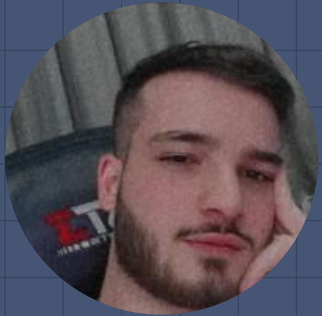
Treinamento e validação de modelos preditivos com AM

GRUPO



Rafael Petry
00314280

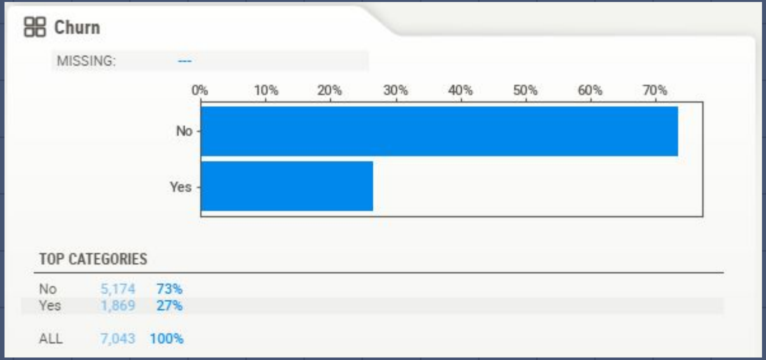


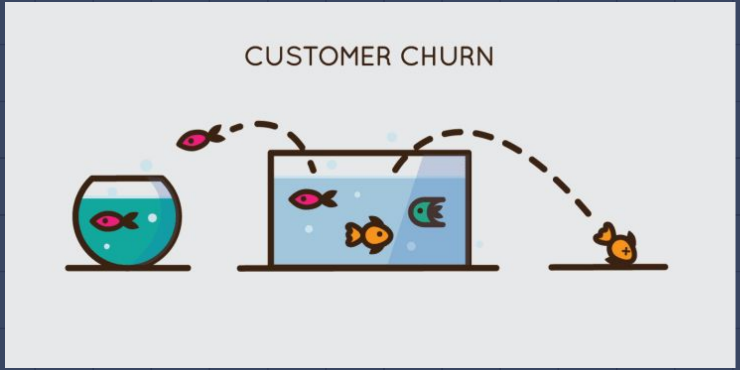
Lucas Melo
00315747

Jean Andrade
00252846

DATASET

Optamos pelo dataset
"Telco Customer Churn" da IBM que contém dados sobre os clientes e serviços contratados





Coluna	Intervalo
gender	[Male, Female]
SeniorCitizen	[0, 1]
Partner	[Yes, No]
Dependents	[Yes, No]
Tenure	[0, ..., 72]
PhotoService	[Yes, No]
MultipleLines	[Yes, No, No service]
InternetService	[DSL, Fiber, No service]
OnlineSecurity	[Yes, No, No service]
OnlineBackup	[Yes, No, No service]

Coluna	Intervalo
DeviceProtection	[Yes, No, No service]
TechSupport	[Yes, No, No service]
StreamingTV	[Yes, No, No service]
StreamingMovies	[Yes, No, No service]
Contract	[Monthly, One Year, Two Years]
PaperlessBilling	[Yes, No]
PaymentMethod	[E-check, Mail-Check, Transfer, Credit Card]
MonthlyCharges	[18.30, ..., 119.00]
*TotalCharges	[18.80, 8684.80]
Churn	[Yes, No]

Qual a probabilidade de um cliente cancelar o serviço?

ATRIBUTOS

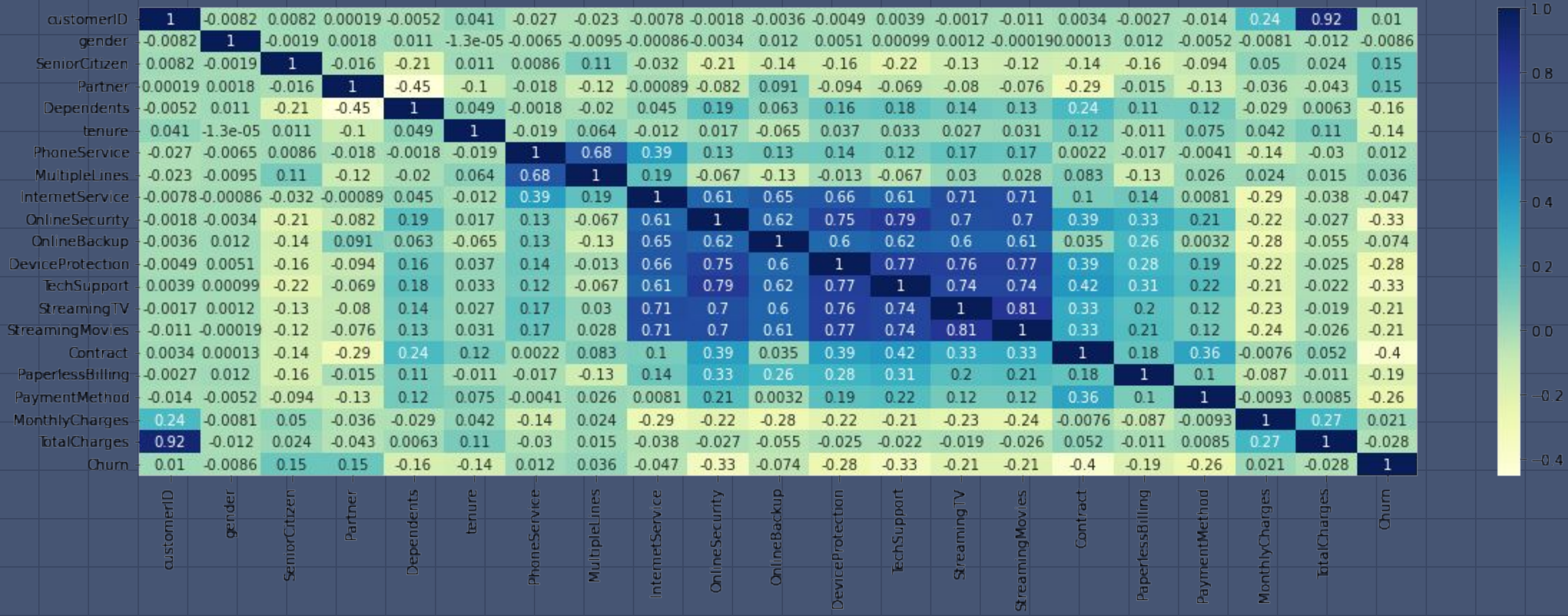
Irrelevantes / Removidos

- customerID - Somente ID do cliente não é importante.
- StreamingTV-Baseado na análise de correlação.

Principais

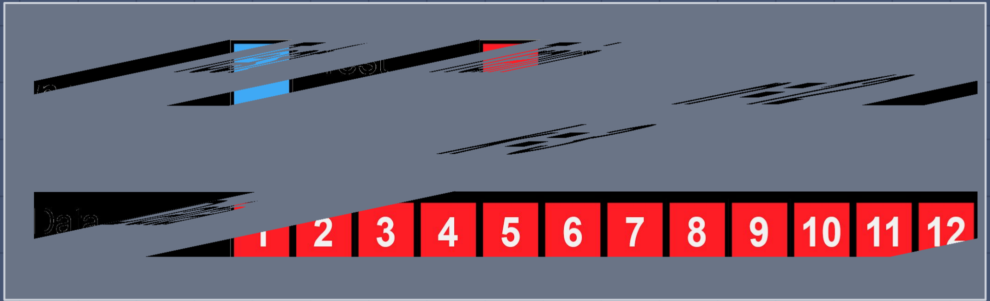
- TotalCharges - Total pago no quarter
- MonthlyCharges - Mensalidade total paga
- Tenure - Total de meses que a pessoa é cliente
- Churn (target - se cliente cancelou)

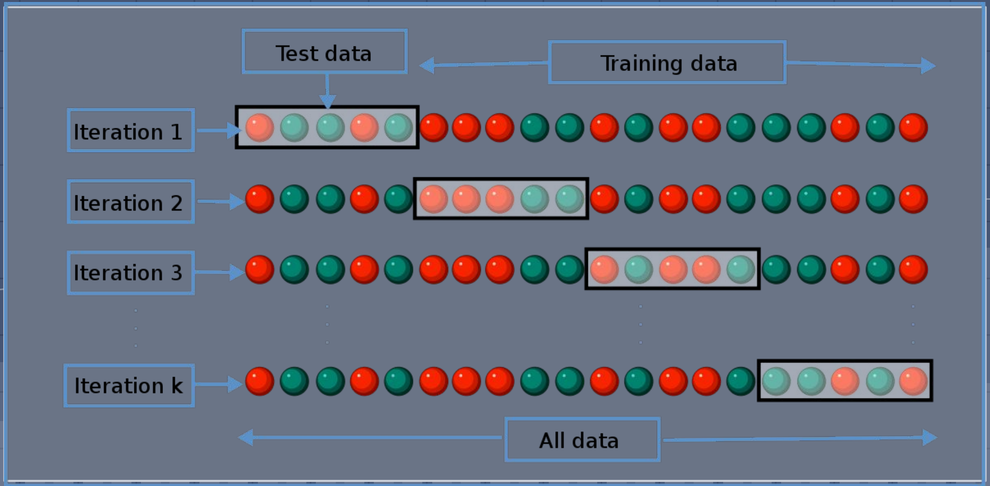
Correlação


customerID 1 -0.0082 0.0082 0.00019 -0.0052 0.041 -0.027 -0.023 -0.0078 -0.0018 -0.0036 -0.0049 0.0039 -0.0017 -0.011 0.0034 -0.0027 -0.014 0.24 0.92 0.01
gender -0.0082 1 -0.0019 0.0018 0.011 -1.3e-05 -0.0065 -0.0095 -0.00086 -0.0034 0.012 0.0051 0.00099 0.0012 -0.00019 0.00013 0.012 -0.0052 -0.0081 -0.012 -0.0086
SeniorCitizen 0.0082 -0.0019 1 -0.016 -0.21 0.011 0.0086 0.11 -0.032 -0.21 -0.14 -0.16 -0.22 -0.13 -0.12 -0.14 -0.16 -0.094 0.05 0.024 0.15
Partner 0.00019 0.0018 -0.016 1 -0.45 -0.1 -0.018 -0.12 -0.00089 -0.082 0.091 -0.094 -0.069 -0.08 -0.076 -0.29 -0.015 -0.13 -0.036 -0.043 0.15
Dependents -0.0052 0.011 -0.21 -0.45 1 0.049 -0.0018 -0.02 0.045 0.19 0.063 0.16 0.18 0.14 0.13 0.24 0.11 0.12 -0.029 0.0063 -0.16
tenure 0.041 -1.3e-05 0.011 -0.1 0.049 1 -0.019 0.064 -0.012 0.017 -0.065 0.037 0.033 0.027 0.031 0.12 -0.011 0.075 0.042 0.11 -0.14
PhoneService -0.027 -0.0065 0.0086 -0.018 -0.0018 -0.019 1 0.68 0.39 0.13 0.13 0.14 0.12 0.17 0.17 0.0022 -0.017 -0.0041 -0.14 -0.03 0.012
MultipleLines -0.023 -0.0095 0.11 -0.12 -0.02 0.064 0.68 1 0.19 -0.067 -0.13 -0.013 -0.067 0.03 0.028 0.083 -0.13 0.026 0.024 0.015 0.036
InternetService -0.0078 -0.00086 -0.032 -0.00089 0.045 -0.012 0.39 0.19 1 0.61 0.65 0.66 0.61 0.71 0.71 0.1 0.14 0.0081 -0.29 -0.038 -0.047
OnlineSecurity -0.0018 -0.0034 -0.21 -0.082 0.19 0.017 0.13 -0.067 0.61 1 0.62 0.75 0.79 0.7 0.7 0.39 0.33 0.21 -0.22 -0.027 -0.33
OnlineBackup -0.0036 0.012 -0.14 0.091 0.063 -0.065 0.13 -0.13 0.65 0.62 1 0.6 0.62 0.6 0.61 0.035 0.26 0.0032 -0.28 -0.055 -0.074
DeviceProtection -0.0049 0.0051 -0.16 -0.094 0.16 0.037 0.14 -0.013 0.66 0.75 0.6 1 0.77 0.76 0.77 0.39 0.28 0.19 -0.22 -0.025 -0.28
TechSupport 0.0039 0.00099 -0.22 -0.069 0.18 0.033 0.12 -0.067 0.61 0.79 0.62 0.77 1 0.74 0.74 0.42 0.31 0.22 -0.21 -0.022 -0.33
StreamingTV -0.0017 0.0012 -0.13 -0.08 0.14 0.027 0.17 0.03 0.71 0.7 0.6 0.76 0.74 1 0.81 0.33 0.2 0.12 -0.23 -0.019 -0.21
StreamingMovies -0.011 -0.00019 -0.12 -0.076 0.13 0.031 0.17 0.028 0.71 0.7 0.61 0.77 0.74 0.81 1 0.33 0.21 0.12 -0.24 -0.026 -0.21
Contract 0.0034 0.00013 -0.14 -0.29 0.24 0.12 0.0022 0.083 0.1 0.39 0.035 0.39 0.42 0.33 0.33 1 0.18 0.36 -0.0076 0.052 -0.4
PaperlessBilling -0.0027 0.012 -0.16 -0.015 0.11 -0.011 -0.017 -0.13 0.14 0.33 0.26 0.28 0.31 0.2 0.21 0.18 1 0.1 -0.087 -0.011 -0.19
PaymentMethod -0.014 -0.0052 -0.094 -0.13 0.12 0.075 -0.0041 0.026 0.0081 0.21 0.0032 0.19 0.22 0.12 0.12 0.36 0.1 1 -0.0093 0.0085 -0.26
MonthlyCharges 0.24 -0.0081 0.05 -0.036 -0.029 0.042 -0.14 0.024 -0.29 -0.22 -0.28 -0.22 -0.21 -0.23 -0.24 -0.0076 -0.087 -0.0093 1 0.27 0.021
TotalCharges 0.92 -0.012 0.024 -0.043 0.0063 0.11 -0.03 0.015 -0.038 -0.027 -0.055 -0.025 -0.022 -0.019 -0.026 0.052 -0.011 0.0085 0.27 1 -0.028
Churn 0.01 -0.0086 0.15 0.15 -0.16 -0.14 0.012 0.036 -0.047 -0.33 -0.074 -0.28 -0.33 -0.21 -0.21 -0.4 -0.19 -0.26 0.021 -0.028 1
customerID gender SeniorCitizen Partner Dependents tenure PhoneService MultipleLines InternetService OnlineSecurity OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies Contract PaperlessBilling PaymentMethod MonthlyCharges TotalCharges Churn
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

K-fold cross validation

1. Divida aleatóriamente os dados em *k* subconjuntos (folds)
2. Use a parte dos dados como treino
3. Use o fold j para teste
4. Repita o processo *k* vezes





K-fold cross validation

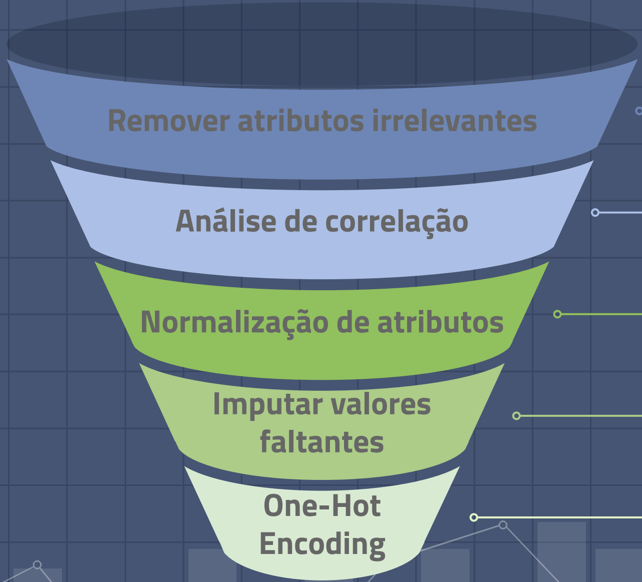
```
def random_k_folds(instance_list: List[Instance], k: int, merge_remainders=True):
    instances = instance_list.copy()
    random.Random(0).shuffle(instances)
    max_fold_size = len(instances) // k
    k_folds = [instances[i * max_fold_size:(i + 1) * max_fold_size] for i in range(k)
    ]
    remainders = instances[k * max_fold_size:]
    return remainders_merge_policy(k_folds, remainders) if merge_remainders else (
    k_folds, remainders)
```

```
def stratified_k_folds(instance_list: List[Instance], k: int, merge_remainders=True):
    instances = instance_list.copy()
    labels = {i.label() for i in instances}
    labeled_instances = {l: [i for i in instances if i.label() == l] for l in labels}
    label_folds = [random_k_folds(labeled_instances[l], k, merge_remainders=False)
    for l in labels]
    labeled_k_folds = [[e for l in f for e in l] for f, r in label_folds]
    k_folds = [[e for l in labeled_k_folds for e in l[i * len(l) // k:(i + 1) * len(l
    ) // k]] for i in range(k)]
    k_folds = [sorted(k, key=operator.attrgetter('_instance_data')) for k in k_folds]
    remainders = [e for f, r in label_folds for e in r]
    return remainders_merge_policy(k_folds, remainders) if merge_remainders else (
    k_folds, remainders)
```

```
class Instance:
    def __init__(self, instance_data: Any, label: Union[str, int]):
        self._instance_data = instance_data
        self._label = label

    def data(self):
        return self._instance_data

    def label(self):
        return self._label

    def __repr__(self):
        return f"{self._instance_data}-{self._label}"
```

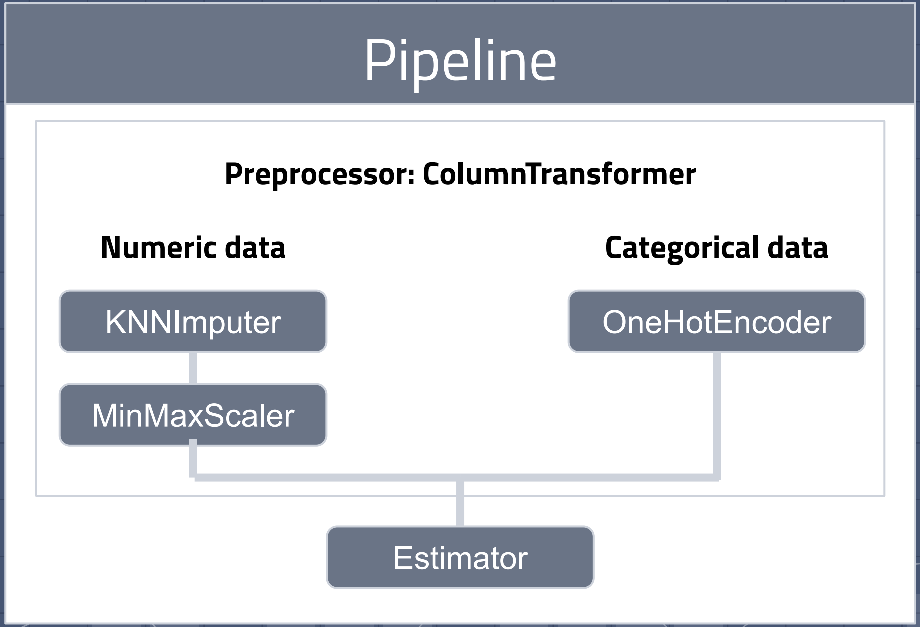
```
def return_dfs(X:pd.DataFrame, y:pd.DataFrame, k:int):
    listt = []
    for i, yy in y.items():
        '''
        cria uma instancia da classe para cada instancia do dataset
        cada instancia possui o index no dataset e a label correspondente
        '''
        listt.append(Instance(i, yy))
    s_folds = stratified_k_folds(listt, k)

    folds = []
    data = []
    labels = []
    for i in range(k):
        '''
        para cada i em k(quantidade de folds) pega o indices que estao no fold
        e utiliza esses indices para montar os fold de dados e de labels
        e em seguida adiciona esse fold criado na lista de todos folds
        '''
        indx = [a._instance_data for a in s_folds[i]]

        folds.append(indx)
        data.append(X.iloc[indx])
        labels.append(y.iloc[indx])

    return data, labels
```

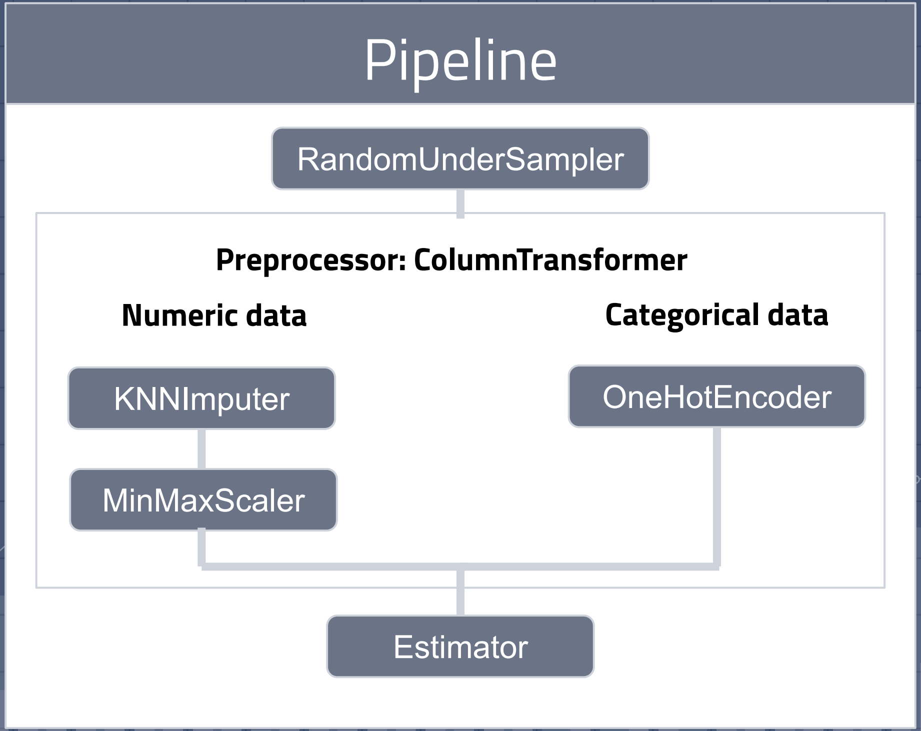
PRÉ PROCESSAMENTO



- Análise empírica
- Análise utilizando matriz de correlação
- Normalização utilizando Min-Max Scaling
- Imputação de valores faltantes usando KNNImputer
- Transforma os valores de um atributo categórico em novos atributos

Pipeline


Pipeline
Preprocessor: ColumnTransformer
Numeric data
Categorical data
KNNImputer
MinMaxScaler
OneHotEncoder
Estimator


Pipeline
RandomUnderSampler
Preprocessor: ColumnTransformer
Numeric data
Categorical data
KNNImputer
MinMaxScaler
OneHotEncoder
Estimator

Pipeline

```
    numeric_features = ["MonthlyCharges", "TotalCharges", "tenure", "gender", "
Partner", "Dependents",
                        "PhoneService", "PaperlessBilling"]
    numeric_transformer = Pipeline(
        steps=[("knnImp", KNNImputer(n_neighbors=3)), ("scaler", MinMaxScaler())]
    )
    categorical_features = ["InternetService", "PaymentMethod", 'MultipleLines',
'OnlineSecurity', 'OnlineBackup',
                            'DeviceProtection', 'TechSupport', 'StreamingMovies',
 'Contract']
    categorical_transformer = Pipeline(
        steps=[("OneHot", OneHotEncoder(handle_unknown="ignore"))]
    )
    preprocessor = ColumnTransformer(
        transformers=[
            ("num", numeric_transformer, numeric_features),
            ("cat", categorical_transformer, categorical_features)
        ]
    )
    if rus:
        pipe = Pipeline(
            steps=[("rus", RandomUnderSampler(random_state=0)), ("pp",
preprocessor), ("clf", clf)]
        )
    else:
        pipe = Pipeline(
            steps=[("pp", preprocessor), ("clf", clf)]
        )

    search = RandomizedSearchCV(pipe, param_grid[i], verbose=True, n_iter=20,
refit=True, random_state=0)
    search.fit(X_train, y_train)

    y_pred = search.predict(X_test)


    y_test = np.array(y_test)

    for metric in ['accuracy', 'precision', 'recall', 'f1measure']:
        if metric not in metadata[run][type(clf).__name__]:
            metadata[run][type(clf).__name__][metric] = {}
    metadata[run][type(clf).__name__]['accuracy'] = accuracy(y_test, y_pred)
    metadata[run][type(clf).__name__]['precision'] = precision(y_test, y_pred)
    metadata[run][type(clf).__name__]['recall'] = recall(y_test, y_pred)
    metadata[run][type(clf).__name__]['f1measure'] = f1Measure(y_test, y_pred)
return metadata
```

PseudoCódigo

Algorithm 1 K-fold Cross Validation

$X, y \leftarrow return\_dfs(data_X, data_y, k)$
$all\_metadata \leftarrow \emptyset$
for $X\_test, y\_test \in X, y$ **do**
    $X\_train \leftarrow X - X\_test$
    $y\_train \leftarrow y - y\_test$
    $metadata \leftarrow Pipeline(X\_train, y\_train, X\_test, y\_test)$
    $all\_metadata \leftarrow all\_metadata \cup metadata$
end for
return $all\_metadata$

ALGORITMOS ESCOLHIDOS

SVM - Support Vector Machine

Algoritmo classificador linear binário não probabilístico. Procura o hiperplano entre dados de duas classes

XGBoost

Algoritmo baseado em árvores de decisão com GradientBoosting.

GaussianNB

Naive Bayes Gaussian.

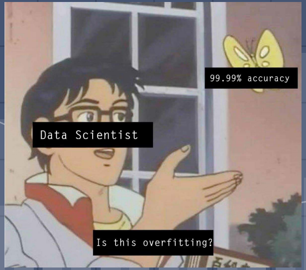
MÉTRICAS

	Desempenho médio	Desvio padrão
Algoritmo 1	10	20
Algoritmo 2	30	15
Algoritmo 3	5	24

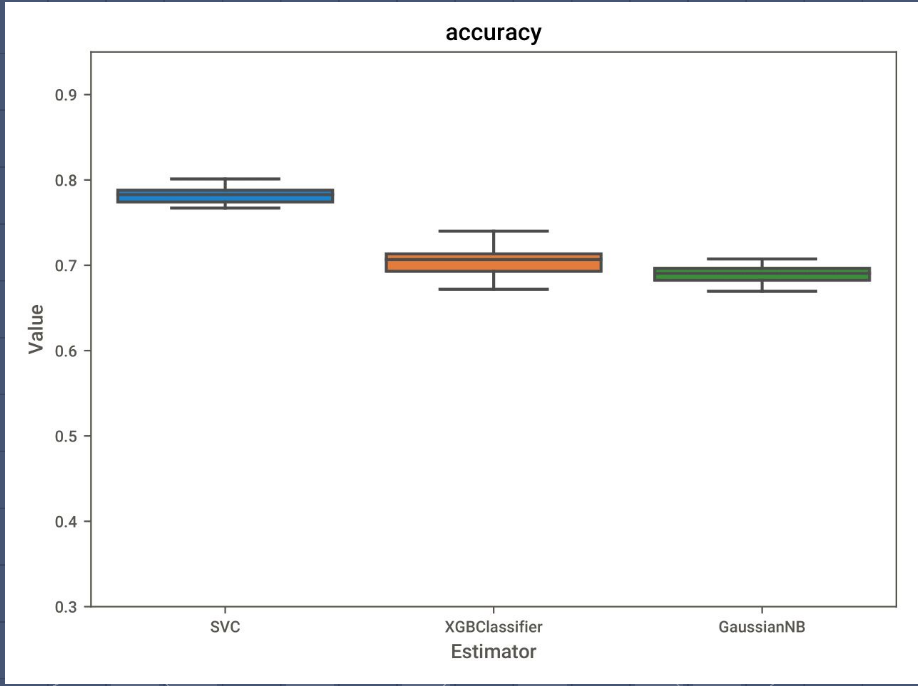
MÉTRICAS (Acurácia e Precisão) - UnderSampling

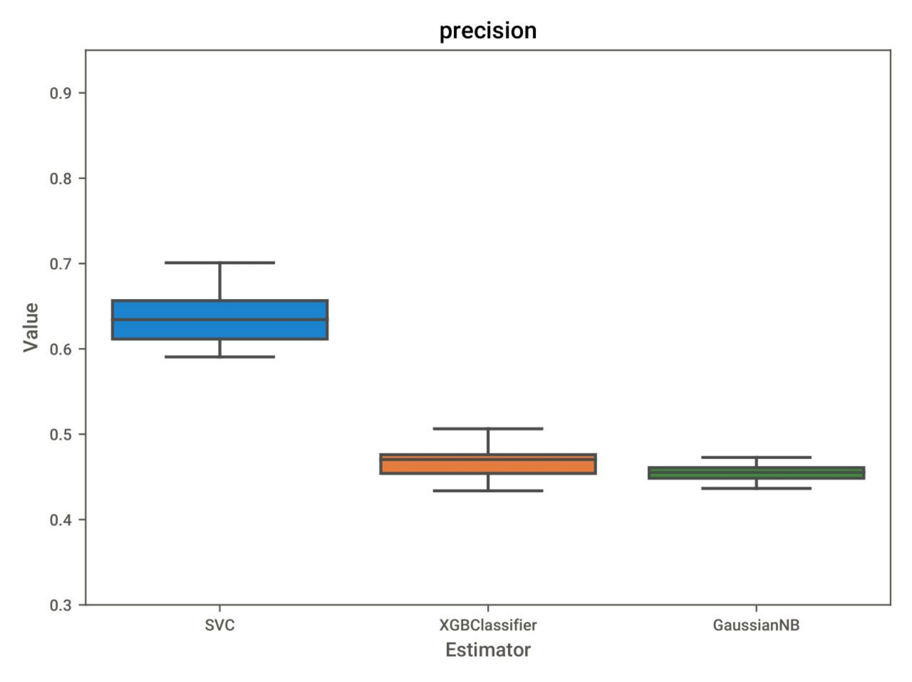
	Desempenho médio	Desvio padrão
SVC	0.782765	0.011507
XGBOOST	0.704531	0.019574
GaussianNB	0.689197	0.012587

	Desempenho médio	Desvio padrão
SVC	0.638734	0.035831
XGBOOST	0.467399	0.020973
GaussianNB	0.454311	0.011943



MÉTRICAS (Acurácia e Precisão) – UnderSampling



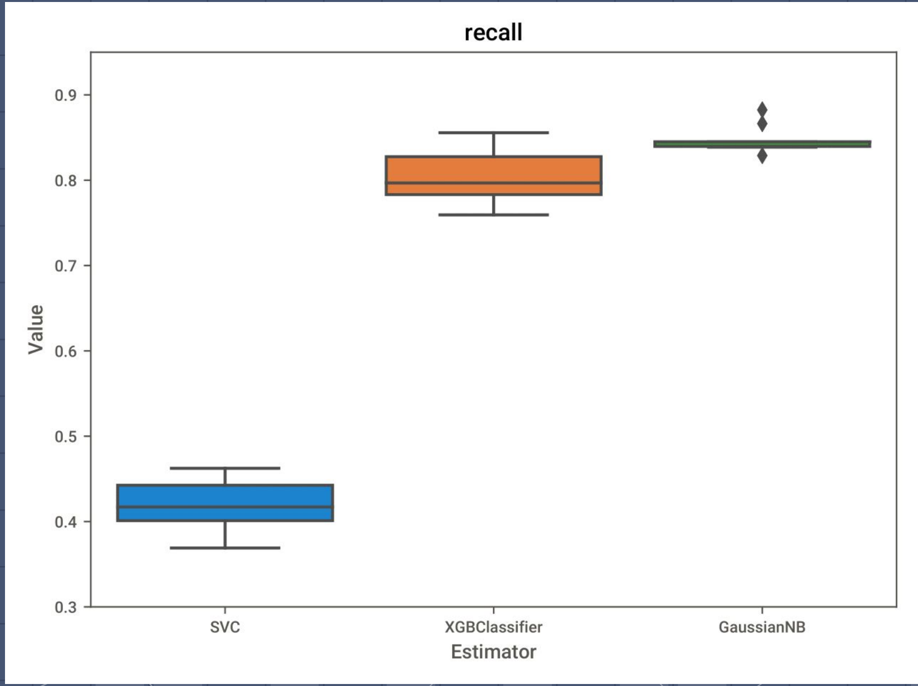


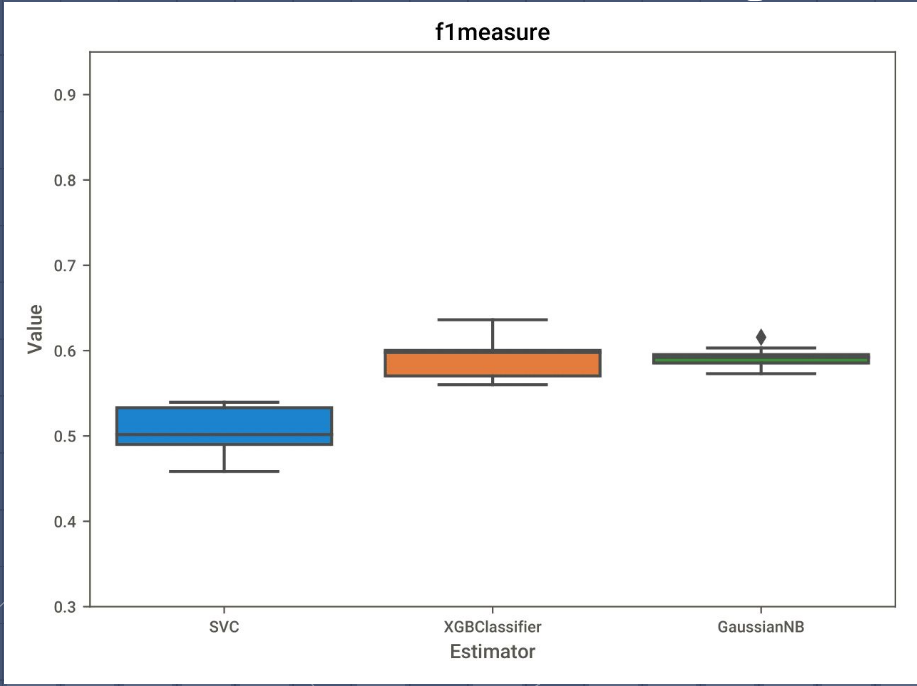
MÉTRICAS (Recall e F1Measure) – UnderSampling

	Desempenho médio	Desvio padrão
SVC	0.419499	0.029269
XGBOOST	0.803631	0.02978
GaussianNB	0.847507	0.015438

	Desempenho médio	Desvio padrão
SVC	0.505901	0.028015
XGBOOST	0.590902	0.023053
GaussianNB	0.591464	0.01246

MÉTRICAS (Recall e F1Measure) – UnderSampling



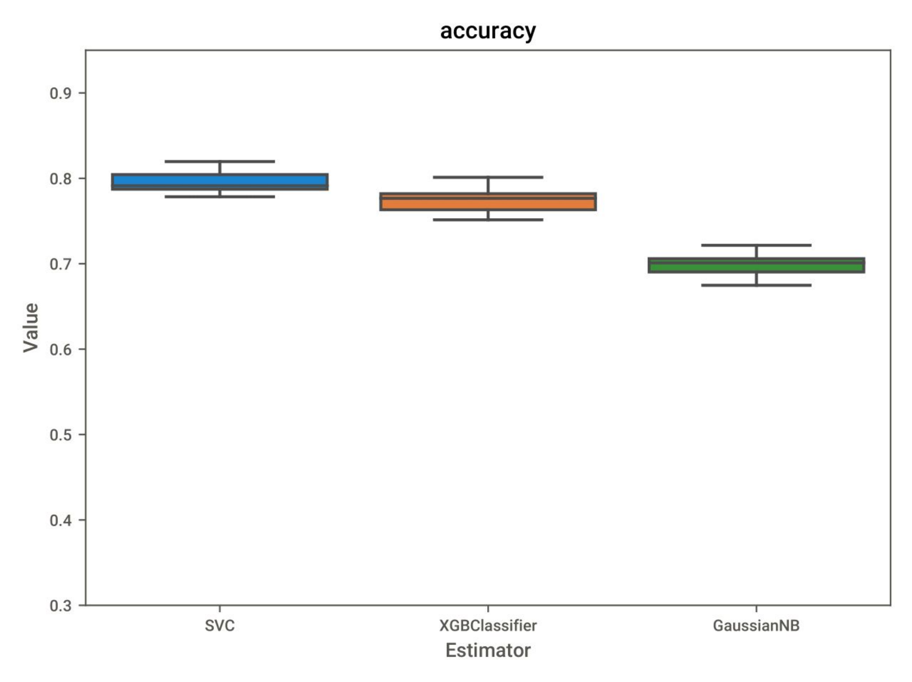


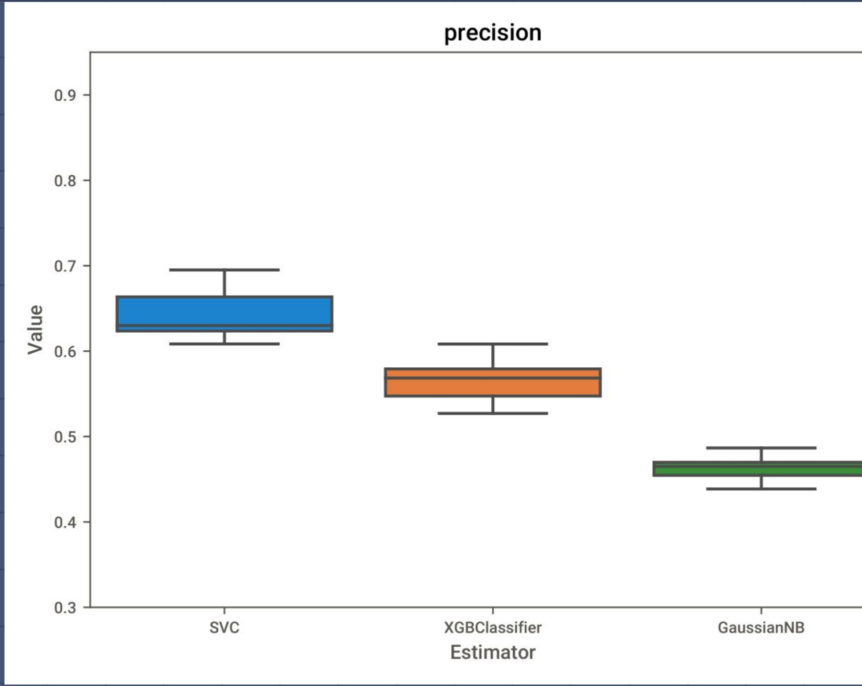
MÉTRICAS (Acurácia e Precisão)

	Desempenho médio	Desvio padrão
SVC	0.795544	0.013279
XGBOOST	0.773534	0.015294
GaussianNB	0.698849	0.013008

	Desempenho médio	Desvio padrão
SVC	0.643444	0.029137
XGBOOST	0.563995	0.02542
GaussianNB	0.46294	0.013204

MÉTRICAS (Acurácia e Precisão)



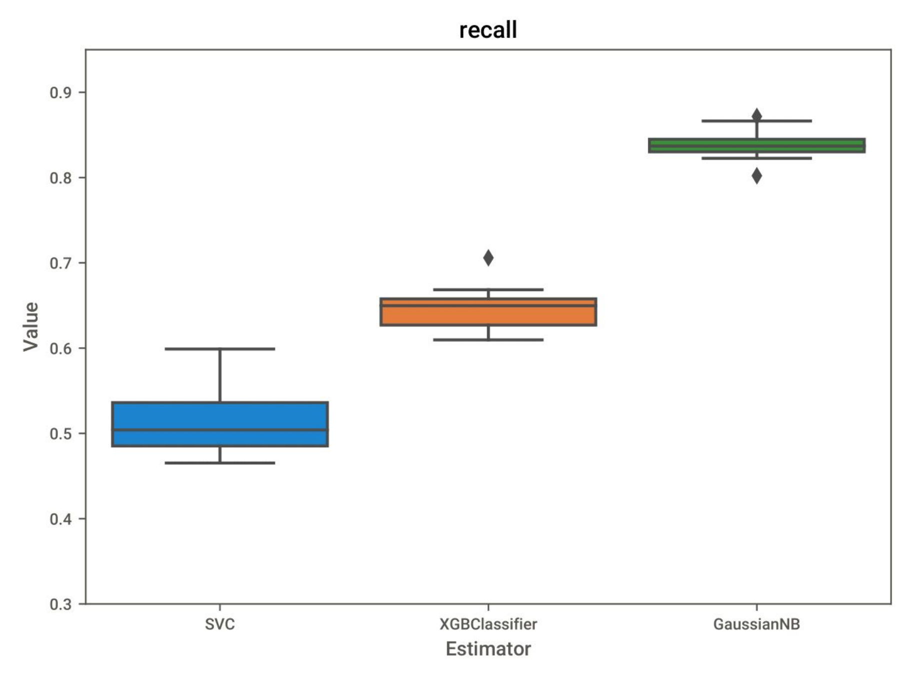


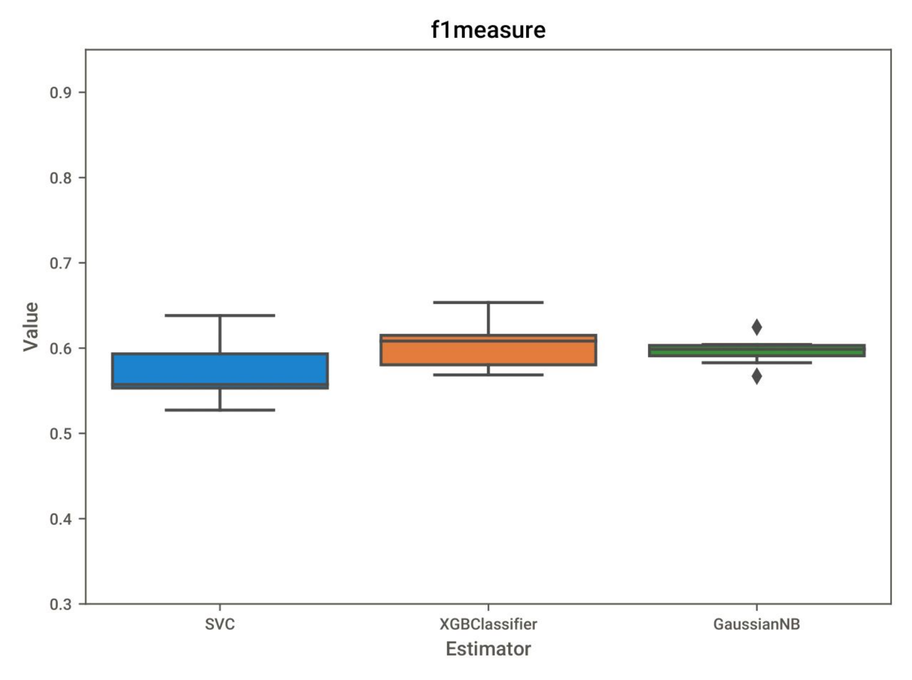
MÉTRICAS (Recall e F1Measure)

	Desempenho médio	Desvio padrão
SVC	0.514174	0.040452
XGBOOST	0.646852	0.028786
GaussianNB	0.838943	0.020164

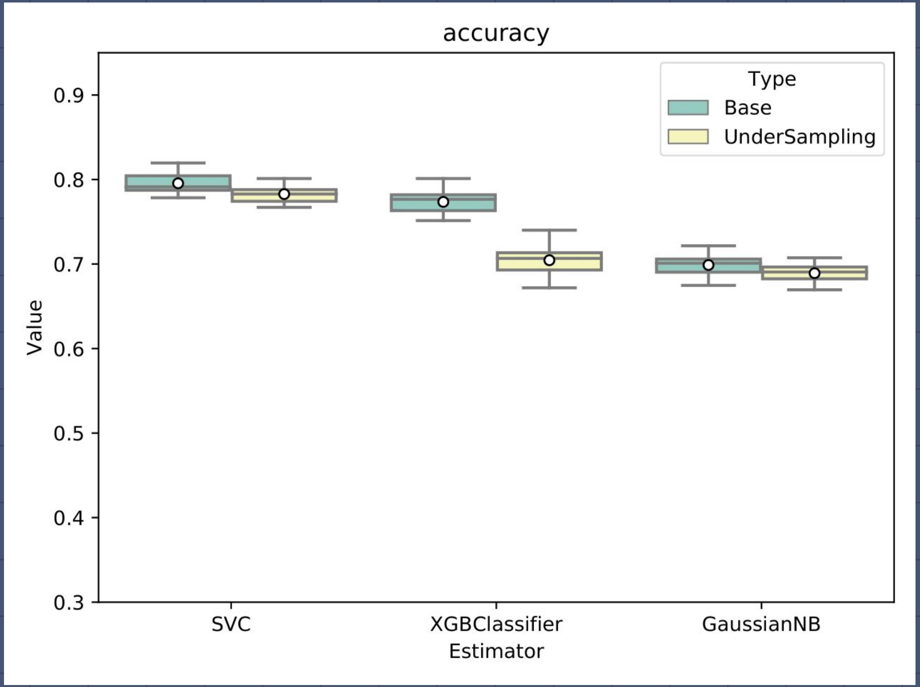
	Desempenho médio	Desvio padrão
SVC	0.571172	0.033598
XGBOOST	0.60254	0.02644
GaussianNB	0.596593	0.015013

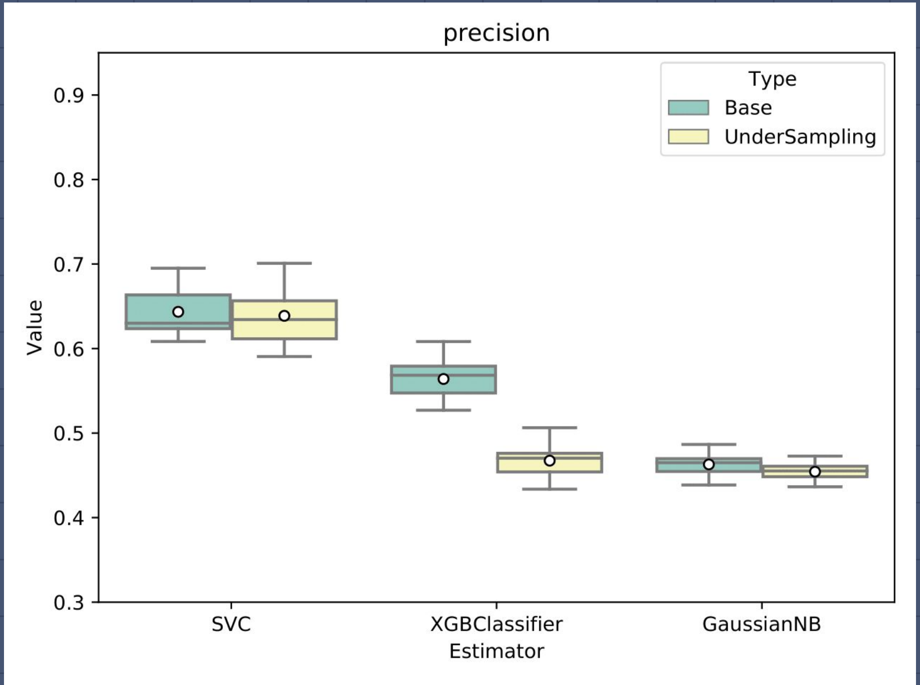
MÉTRICAS (Recall e F1Measure)



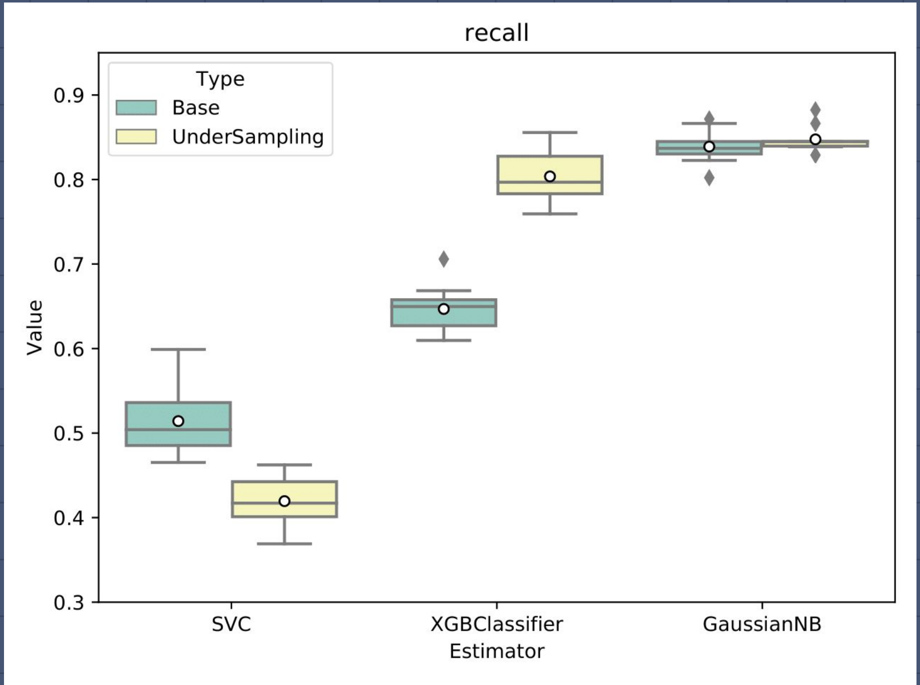


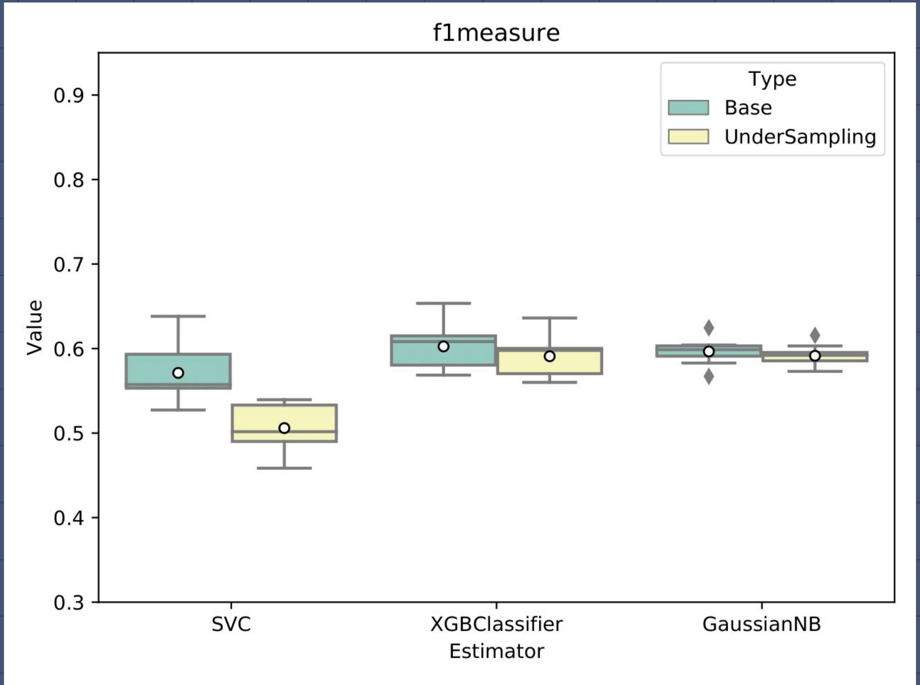
MÉTRICAS (Acurácia e Precisão) – Comparação





MÉTRICAS (Recall e F1Measure) – Comparação





Conclusão

Como o *churn* gera impacto de caráter econômico, acreditamos que a melhor opção de modelo, para este caso, seria o Gaussian Naïve Bayes, pois apresentou o melhor desempenho para **F-measure** e **Recall**.

Acreditamos que a priorização do **Recall** é interessante por priorizar manter os clientes da empresa. E a da **F-measure** por permitir ajuste de parâmetros baseado em características do mercado como custo/retorno de prevenção de *churn*, pois o quanto é interessante poupar caixa da empresa (**reduzir FP**) ou aumentar a arrecadação mantendo clientes (**reduzir FN**) pode variar de acordo com as condições de mercado e custos de aquisição ou manutenção de cliente.

Obrigado!

Perguntas?