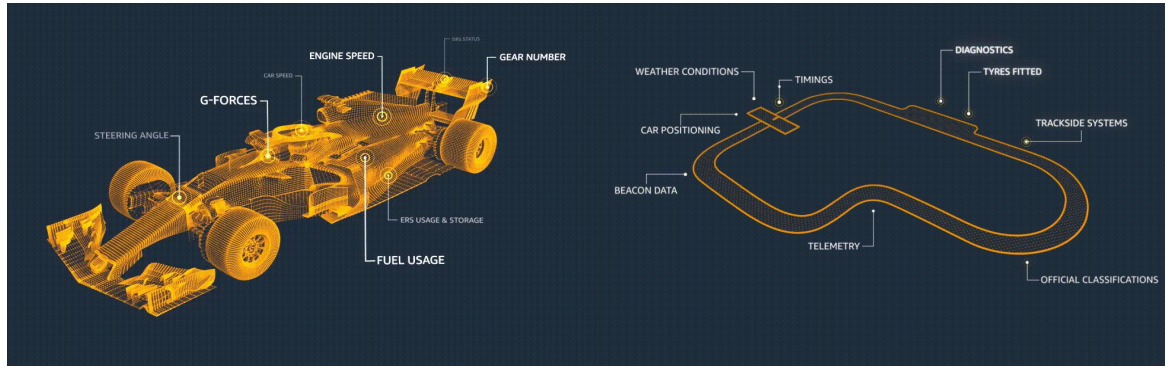
PREDIÇÃO DO CAMPEONATO DE FÓRMULA 1

Mateus Guilherme Fuini
Michelle Melo Cavalcante

FÓRMULA 1 É ORIENTADO POR DADOS

Campeonato Mundial com uma série de eventos de automobilismo em 21 países em que pilotos profissionais correm em carros de assento único em pistas personalizadas ou em percursos urbanos em busca do título do Campeonato Mundial.

As corridas de Fórmula 1 têm mais de **500 milhões de fãs globais** e geraram **USD 1,8 bilhão em receita total em 2017**.



https://aws.amazon.com/pt/f1/

Estratégia de corrida

Usando dados de tempo, a F1 é capaz de criar percepções visuais que permitem aos fãs analisar objetivamente a performance, a estratégia e as táticas de cada equipe e piloto que irão afetar o resultado geral da corrida

Análise de competições

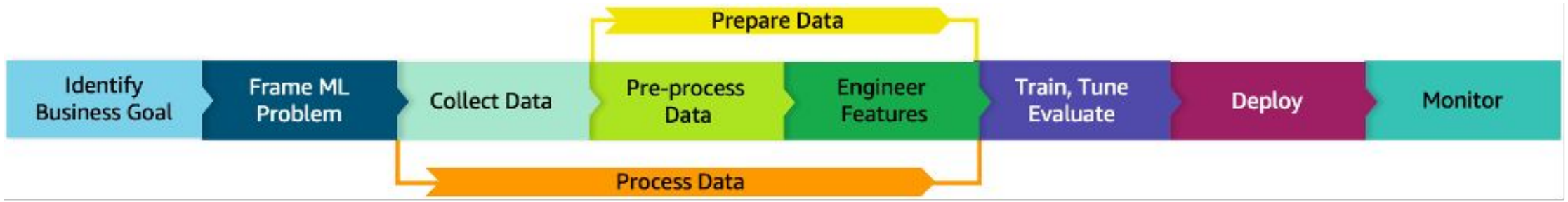
A análise de dados permite à F1 comparar a performance de determinados carros, equipes e pilotos em qualquer parâmetro relevante e classificá-los visualmente para educar os fãs

Performance dos carros

A F1 analisa de perto a aerodinâmica, a performance dos pneus, a unidade de potência, a dinâmica do veículo e a otimização do veículo para oferecer insights que ajudam os fãs a interpretar a performance geral do carro

WELL-ARCHITECTED MACHINE LEARNING

...

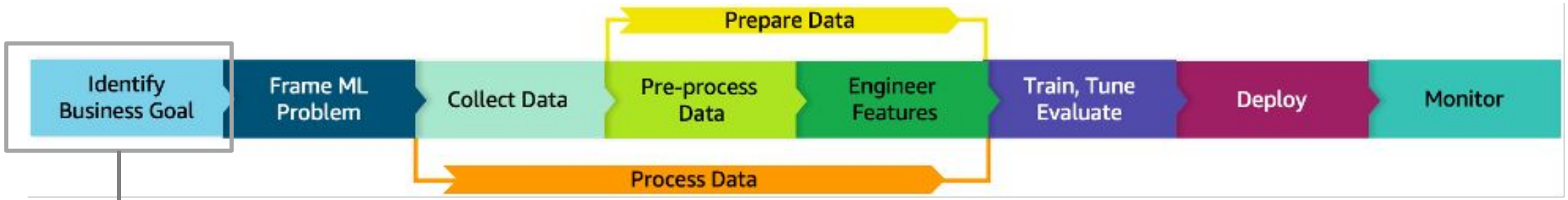


Well-Architected são práticas recomendadas examinadas em cada um dos cinco pilares de **excelência operacional**, **segurança**, **confiabilidade**, **eficiência de desempenho** e **otimização de custos**.

Machine Learning Lifecycle phases

IDENTIFICAÇÃO DOS OBJ. DE NEGÓCIO

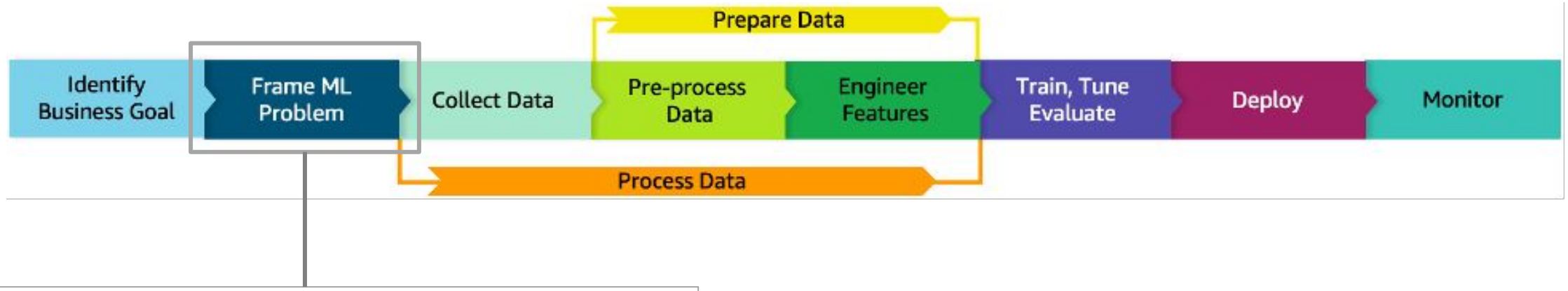
...



Ao obter dados históricos e usá-los para ensinar algoritmos de machine learning complexos, a F1 pode prever os resultados da estratégia de corrida com maior precisão para equipes, carros e pilotos.

Esses modelos são então capazes de prever cenários futuros usando dados históricos conforme as corridas se desdobram.

ENQUADRAMENTO DO PROBLEMA

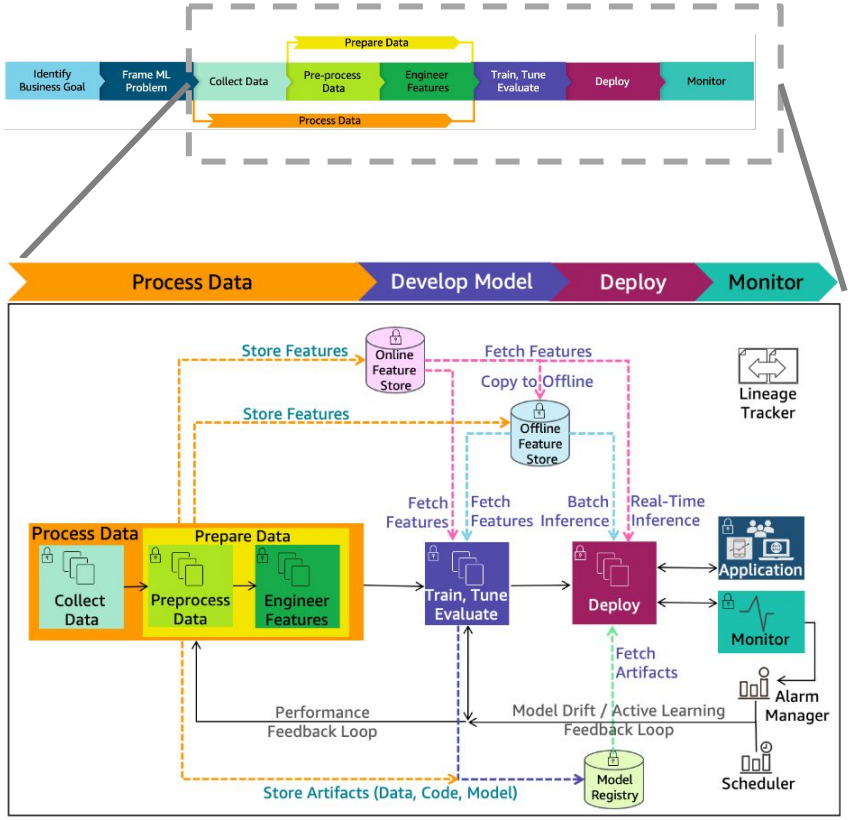


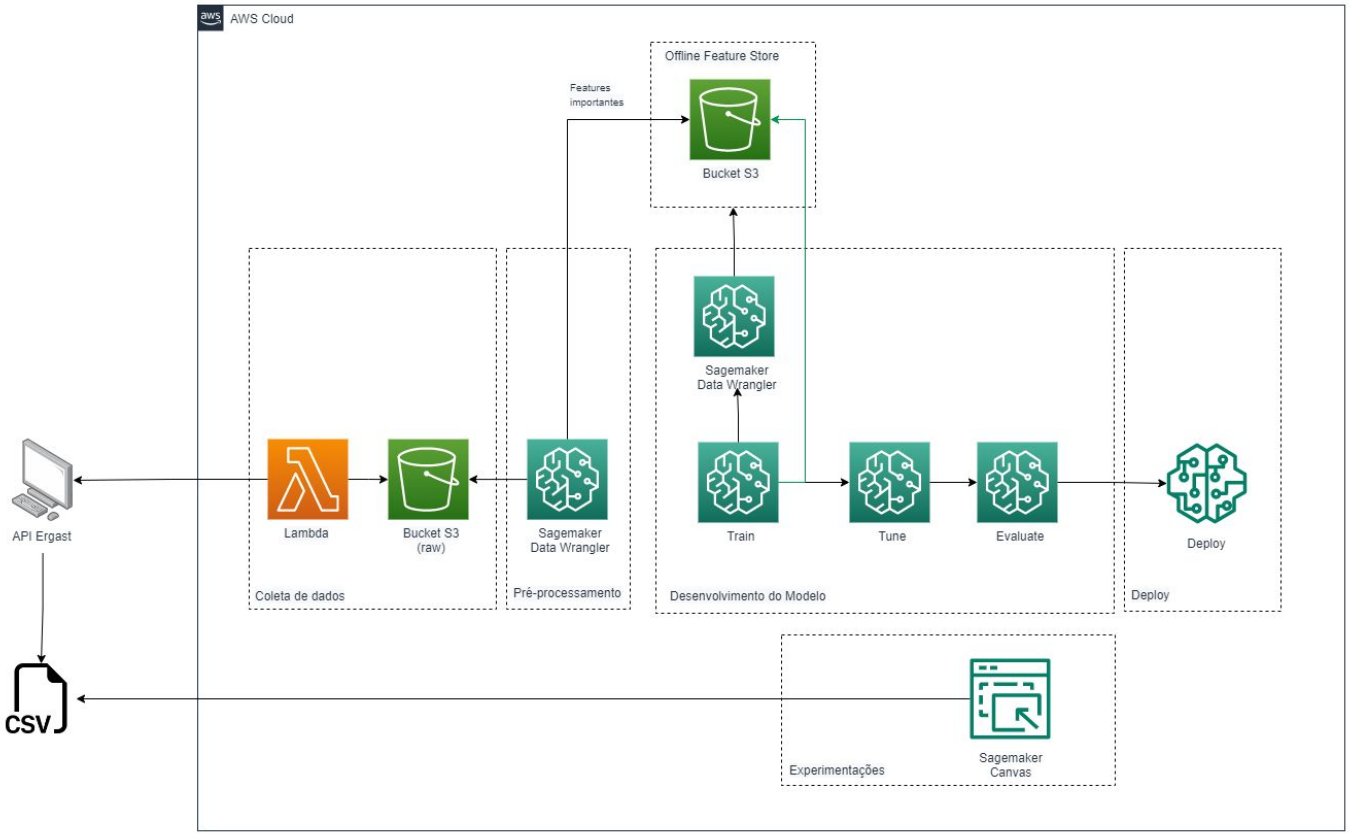
Temos um cenário em que queremos verificar os melhores pilotos e equipes de F1. Atingir essa meta de negócios depende parcialmente da análise histórica do desempenho desses pilotos e equipes.

Neste cenário, vamos **prever o resultado de futuras corridas** (GP ABU DHABI que acontecerá em 20/11/2022) com base em corridas e performances anteriores.

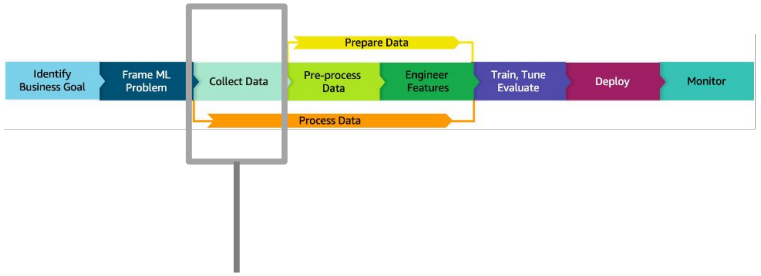
ENQUADRAMENTO DO PROBLEMA

...

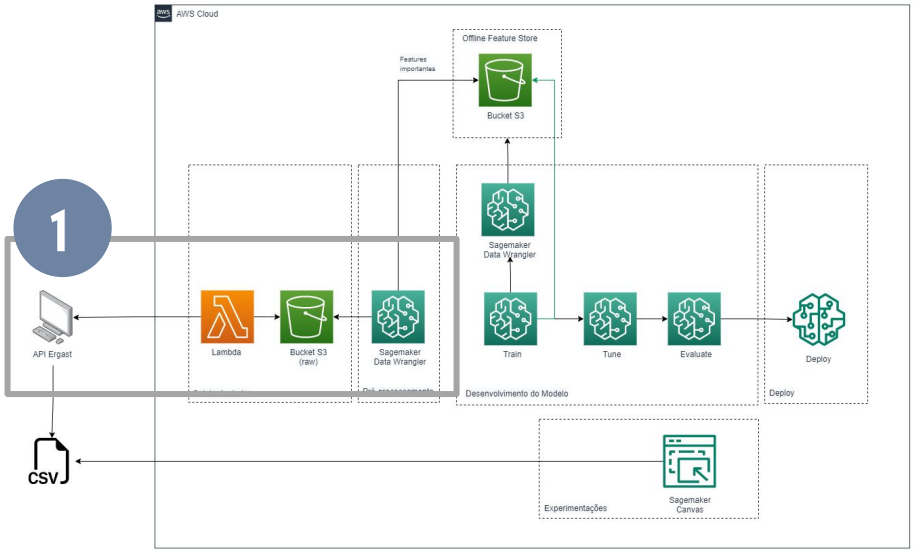




PROCESSAMENTO DE DADOS

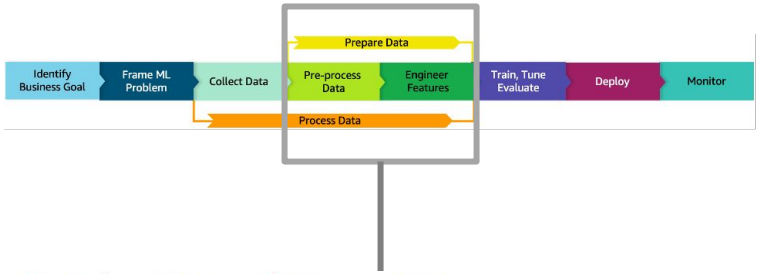


1



1. Coleta de dados .json da Ergast
2. Armazena em Bucket S3
3. Data Wrangler para abstrair boto3 e realizar leitura .json

PROCESSAMENTO DE DADOS



2

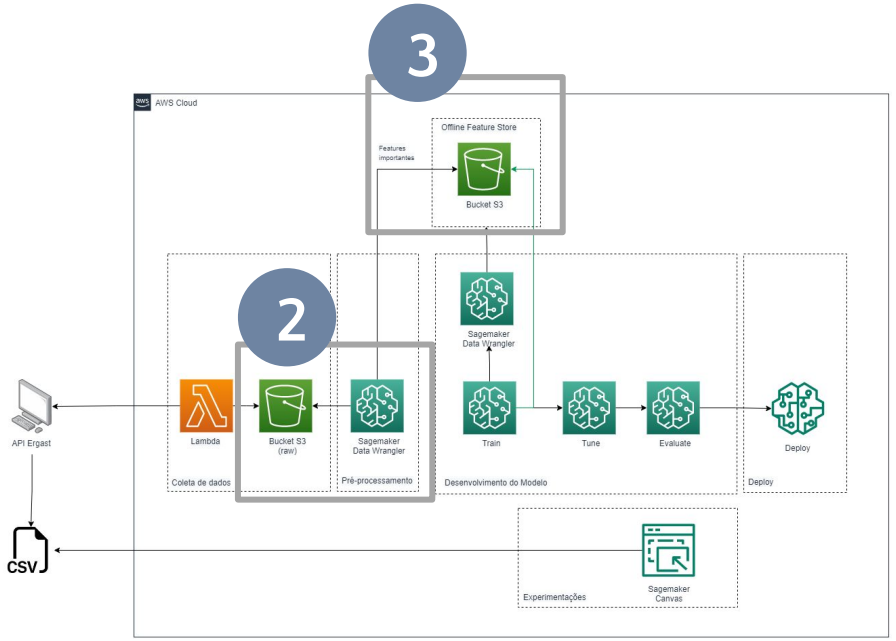
```
RangeIndex: 25367 entries, 0 to 25366
Data columns (total 23 columns):
 #   Column                  Non-Null Count
---  ------                  --------------
 0   year                    25367 non-null
 1   date                    25367 non-null
 2   quali_pos               25367 non-null
 3   position                14953 non-null
 4   points                  25325 non-null
 5   statusId                25367 non-null
 6   dob                     25367 non-null
 7   driver_nationality      25367 non-null
 8   constructor             25367 non-null
 9   constructor_nationality 25367 non-null
 10  GP_name                 25367 non-null
 11  country                 25367 non-null
 12  driver                  25367 non-null
 13  age_at_gp_in_days       25367 non-null
 14  driver_home             25367 non-null
 15  constructor_home        25367 non-null
 16  driver_dnf              25367 non-null
 17  constructor_dnf         25367 non-null
 18  weather_warm            25367 non-null
 19  weather_cold            25367 non-null
 20  weather_dry             25367 non-null
 21  weather_wet             25367 non-null
 22  weather_cloudy          25367 non-null
dtypes: float64(2), int64(13), object(8)
memory usage: 4.5+ MB
```

3

```python
import os
import boto3

bucket = 'feature-store-f1'
prefix = ''

# Upload - S3 bucket
s3_input_data_filtered = (boto3.Session().
                          resource("s3").
                          Bucket(bucket).
                          Object(os.path.join(prefix, "data_filtered.csv")).
                          upload_file("./datasets/data_filtered.csv"))
```

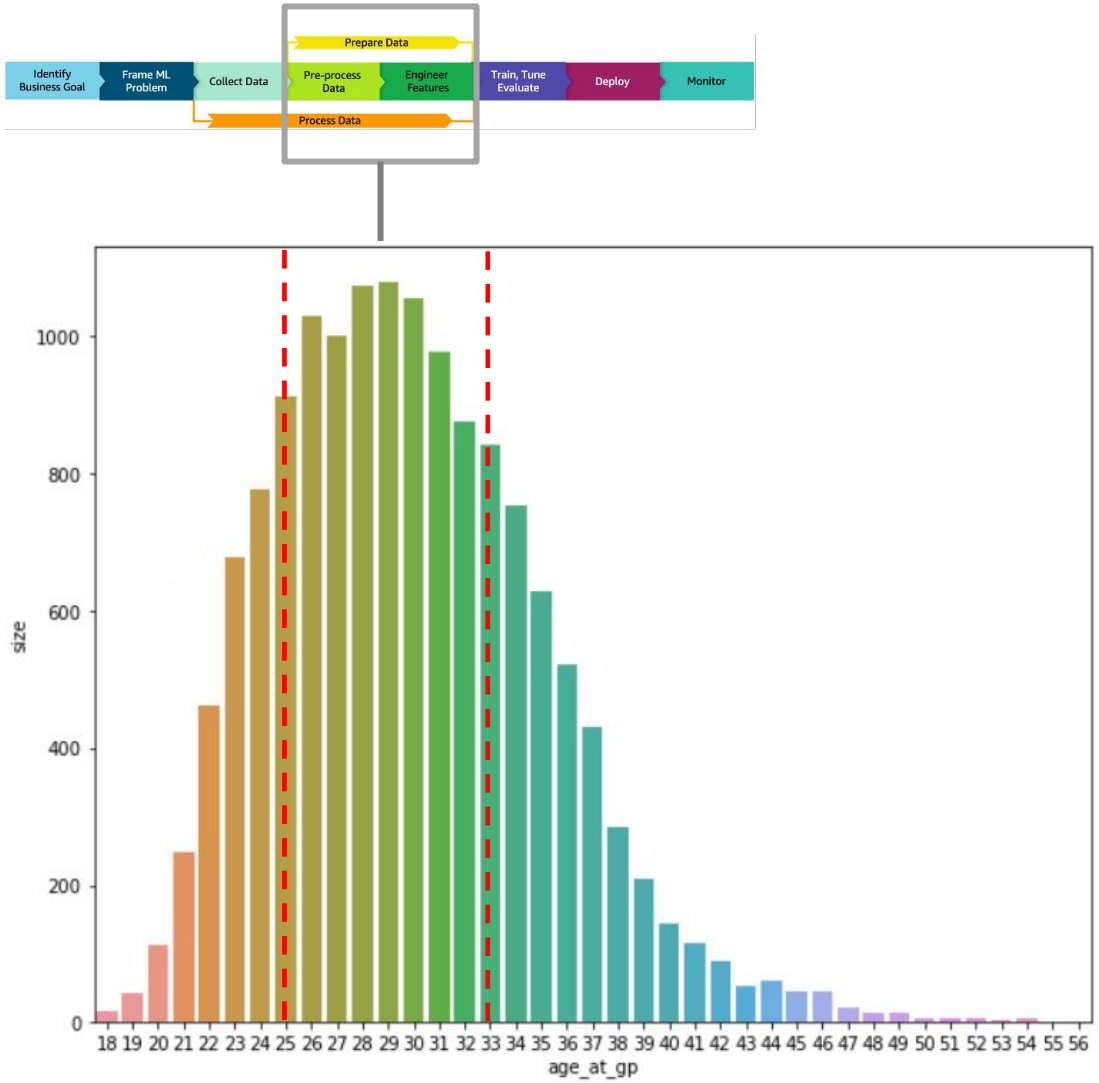
data_filtered.csv

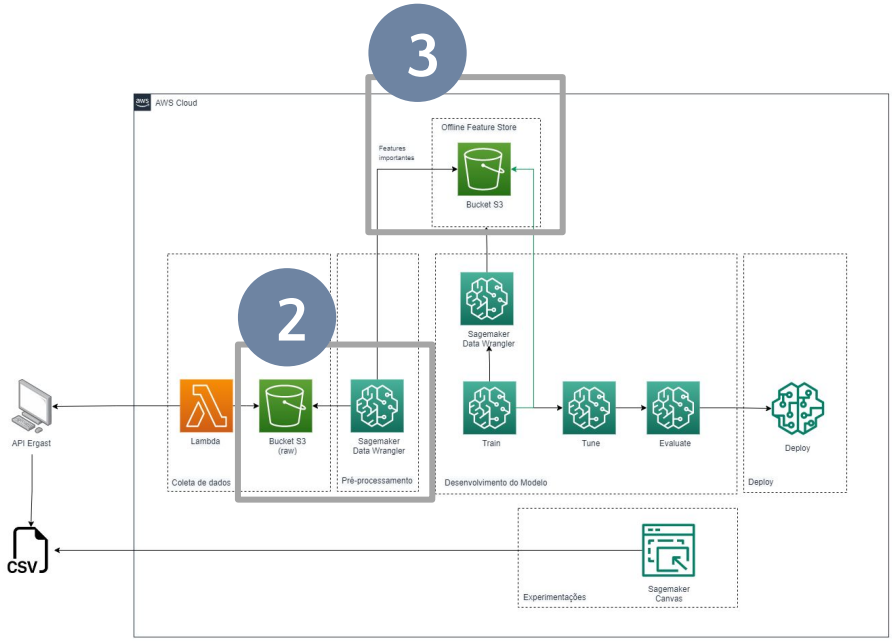
3



1. Análise inicial e merge de todas as tabelas
2. Limpeza dos dados e renomeação de colunas
3. Inclusão de novas features e padronização de atributos
4. Feature Store para armazenamento dos filtros
5. Análise exploratória mais avançada

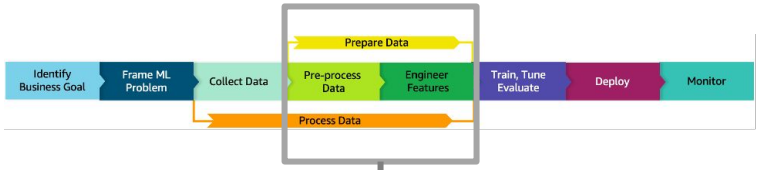
PROCESSAMENTO DE DADOS



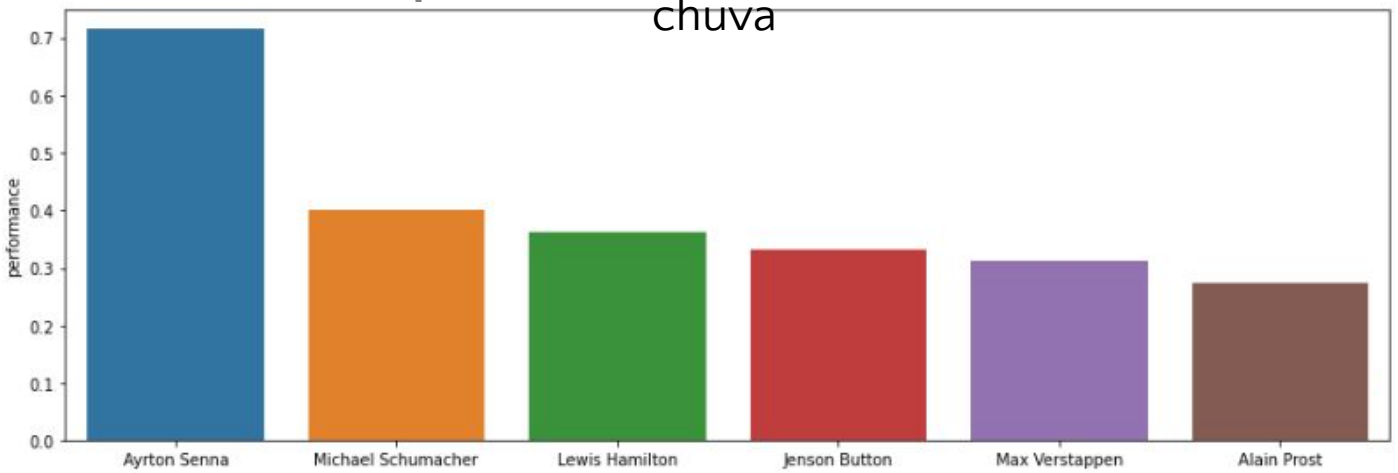


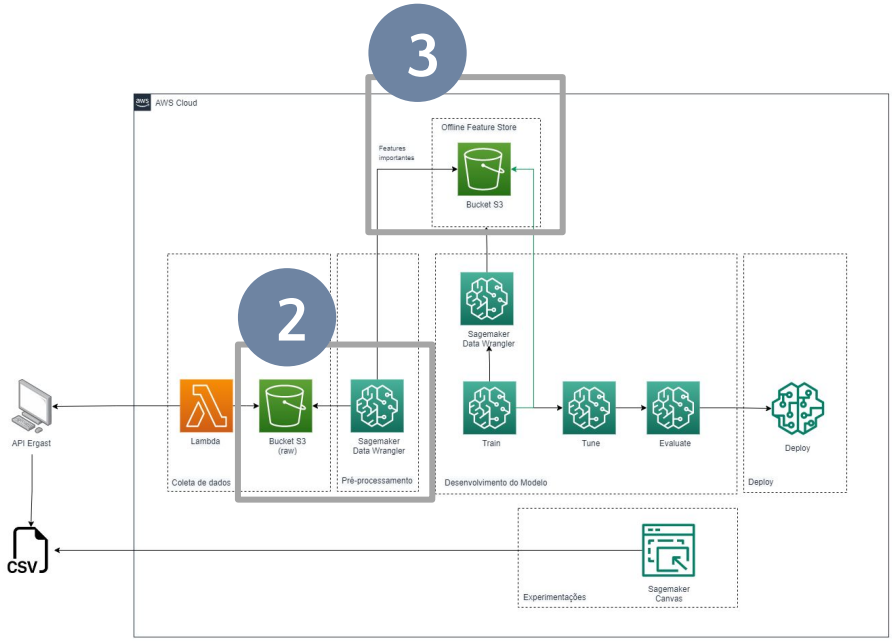
1. Análise inicial e merge de todas as tabelas
2. Limpeza dos dados e renomeação de colunas
3. Inclusão de novas features e padronização de atributos
4. Feature Store para armazenamento dos filtros
5. Análise exploratória mais avançada

PROCESSAMENTO DE DADOS



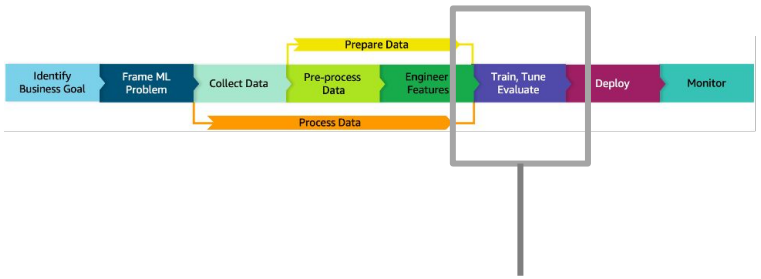
Pilotos com mais vitórias com chuva



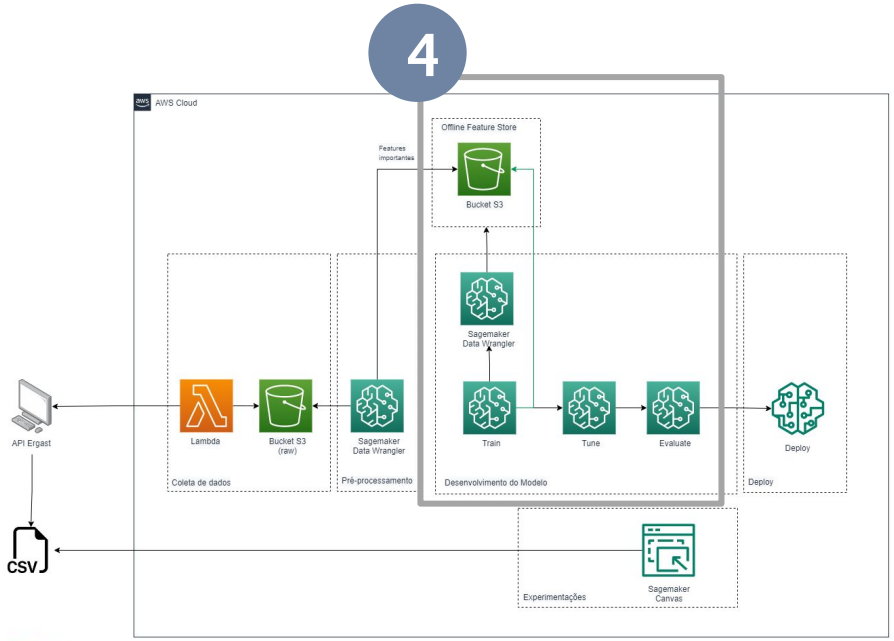


1. Análise inicial e merge de todas as tabelas
2. Limpeza dos dados e renomeação de colunas
3. Inclusão de novas features e padronização de atributos
4. Feature Store para armazenamento dos filtros
5. Análise exploratória mais avançada

DESENVOLVIMENTO DO MODELO

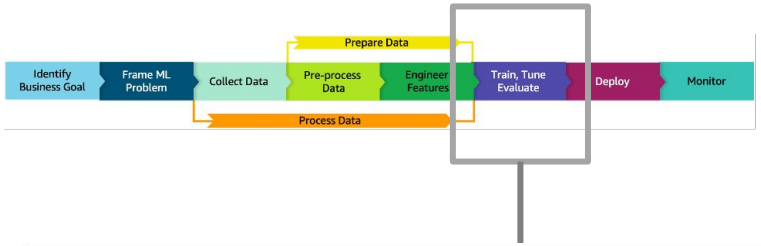


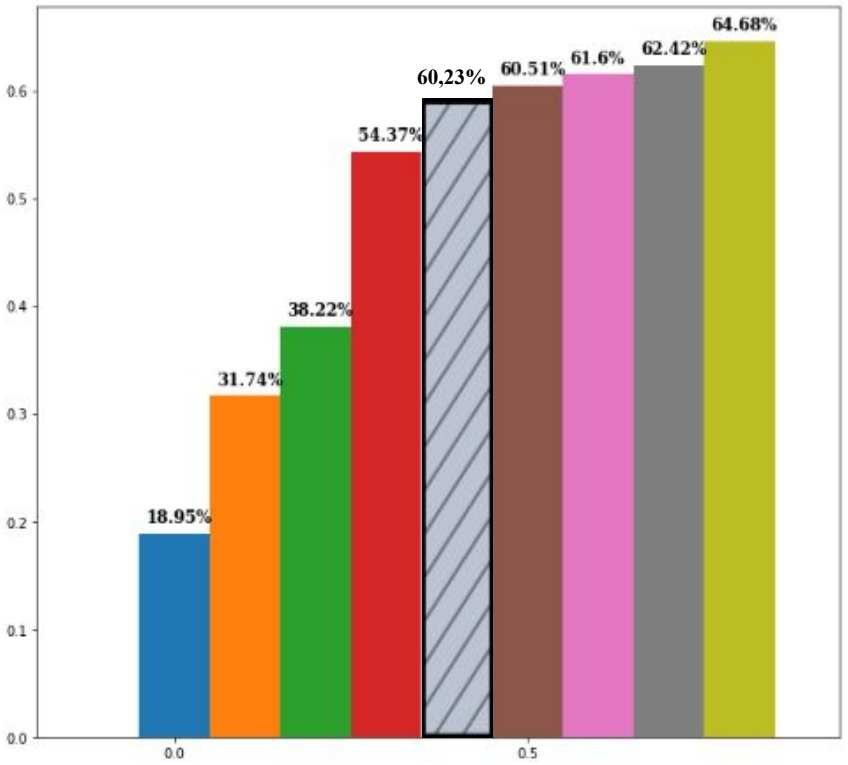

```
from sklearn.model_selection import train_test_split
df_train, df_test, x_train, y_test =
train_test_split(df_tt,df_tt['position'], test_size=0.2, random_state=42)

df_train_c, df_test_c =
train_test_split(df_tt_c, stratify=df_tt_c['position'], test_size=0.2, random_state=42)

sc = StandardScaler()
le = LabelEncoder()
df_train_c['GP_name'] = le.fit_transform(df_train_c['GP_name'])
df_train_c['driver'] = le.fit_transform(df_train_c['driver'])
df_train_c['constructor'] = le.fit_transform(df_train_c['constructor'])
```



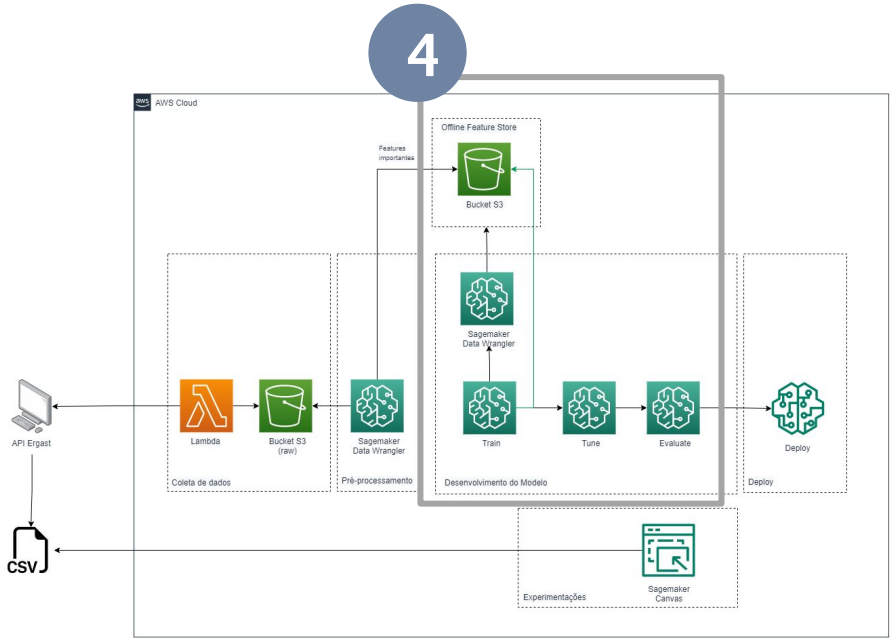
1. **Treinamento do modelo**
2. Ajuste do modelo
3. Avaliação do modelo

DESENVOLVIMENTO DO MODELO



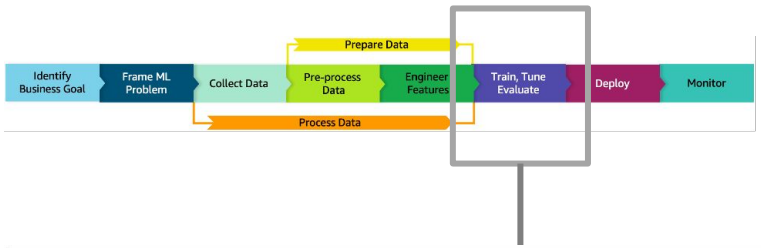


DummyClassifier
SVC
KNeighborsClassifier
DecisionTreeClassifier
XGBClassifier
RandomForestClassifier
LogisticRegression
LGBMClassifier
GradientBoostingClassifier



1. **Treinamento do modelo**
2. Ajuste do modelo
3. Avaliação do modelo

DESENVOLVIMENTO DO MODELO



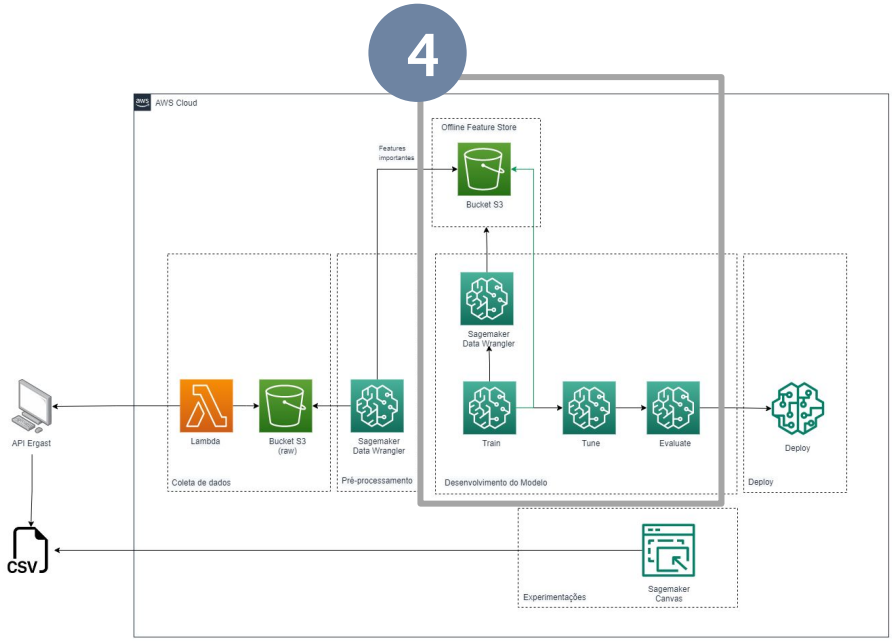

```
rf_rand = GradientBoostingClassifier()
rf_random = RandomizedSearchCV(estimator=rf_rand, param_distributions=random_parms, n_iter=1, cv=10, verbose=2,
                               n_jobs=-1)
rf_random.fit(X, y)
rf_random.best_params_
```

```
Fitting 10 folds for each of 1 candidates, totalling 10 fits
{'n_estimators': 1200,
 'min_samples_split': 10,
 'min_samples_leaf': 2,
 'max_features': 'sqrt',
 'max_depth': 50}
```

best parameters - fitting 10 folds for each of 100 candidates, tottaling 1000 fits {'n_estimators': 400, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 80}

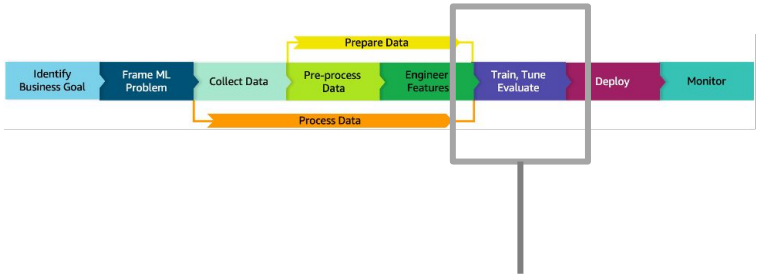
```
rf = GradientBoostingClassifier(n_estimators=400, min_samples_split=5, min_samples_leaf=1, max_features='auto',
                                max_depth=80)
kf = StratifiedKFold(n_splits=10, random_state=None, shuffle=False)
for train_index, test_index in kf.split(X, y):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
cnf_mat_rf = confusion_matrix(y_test, y_pred_rf)
cnf_mat_rf = cnf_mat_rf / cnf_mat_rf.sum()
cnf_mat_rf
```

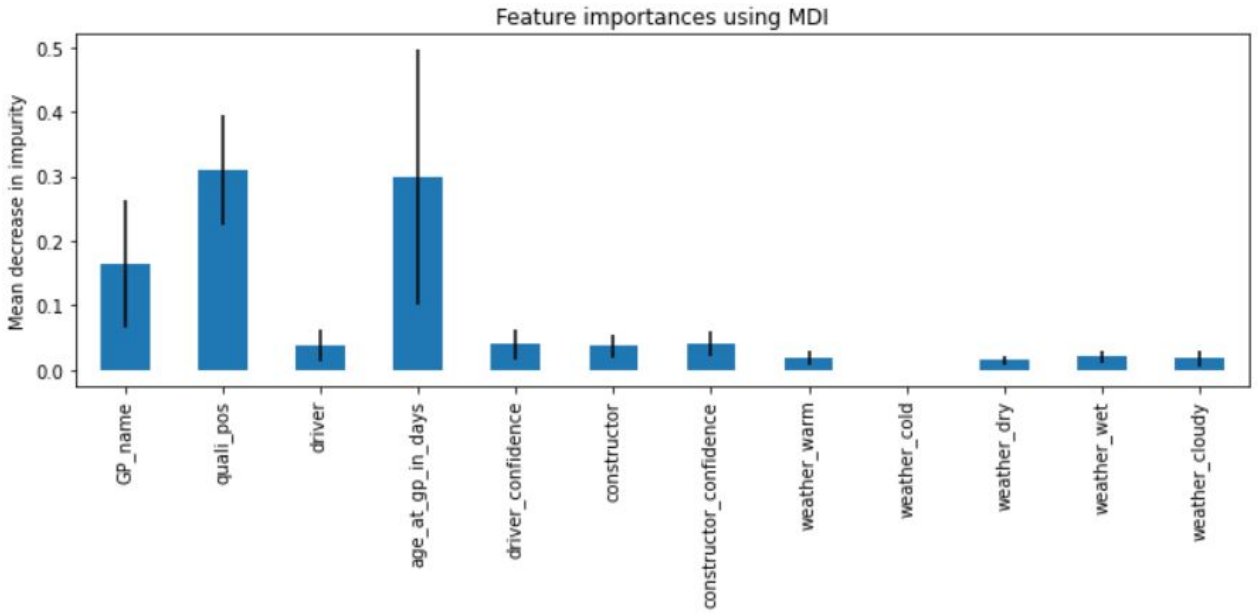
```
array([[0.10566038, 0.10188679, 0.01509434],
       [0.09433962, 0.36226415, 0.09056604],
       [0.00377358, 0.06792453, 0.15849057]])
```

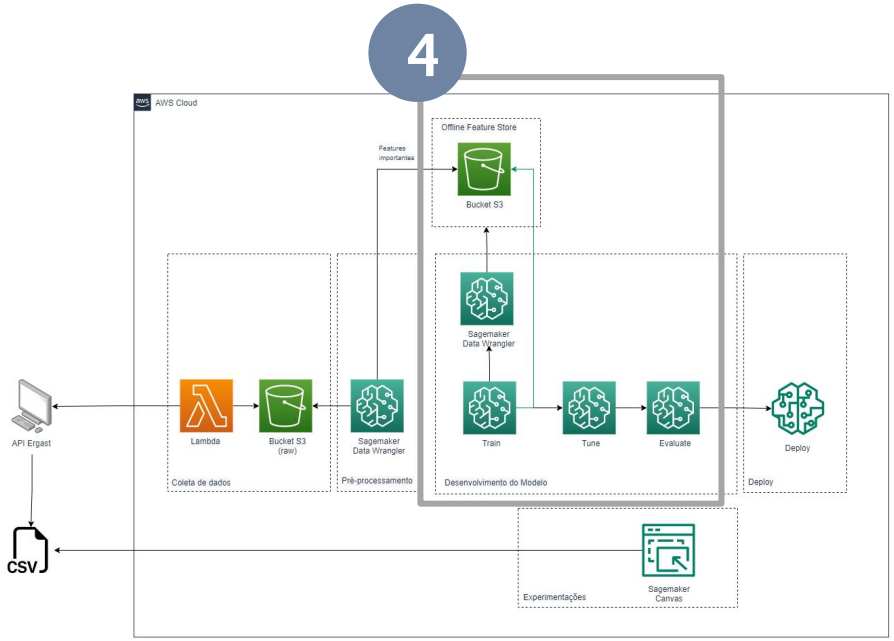


1. Treinamento do modelo
2. **Ajuste do modelo**
3. Avaliação do modelo

DESENVOLVIMENTO DO MODELO

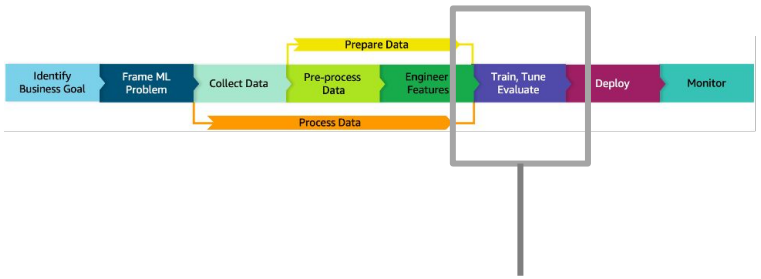






1. Treinamento do modelo
2. Ajuste do modelo
3. **Avaliação do modelo**

DESENVOLVIMENTO DO MODELO

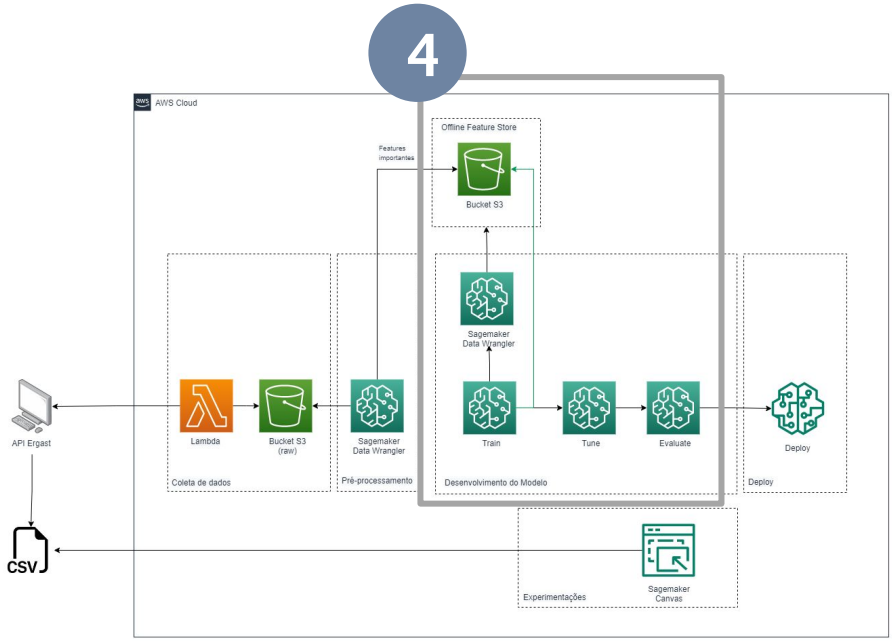



```
def predict_position(circuit, weather):
```

Driver	Prediction
Max Verstappen	0.777143
Charles Leclerc	0.520000
George Russell	0.480000
Lewis Hamilton	0.443810
Carlos Sainz	0.411429

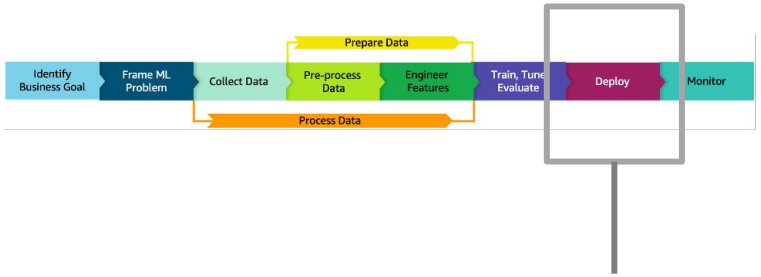
Resultados

Pos	Piloto	
1	M. Verstappen	Red Bull · #1
2	C. Leclerc	Ferrari · #16
3	S. Perez	Red Bull · #11
4	C. Sainz Jr.	Ferrari · #55
5	G. Russell	Mercedes · #63



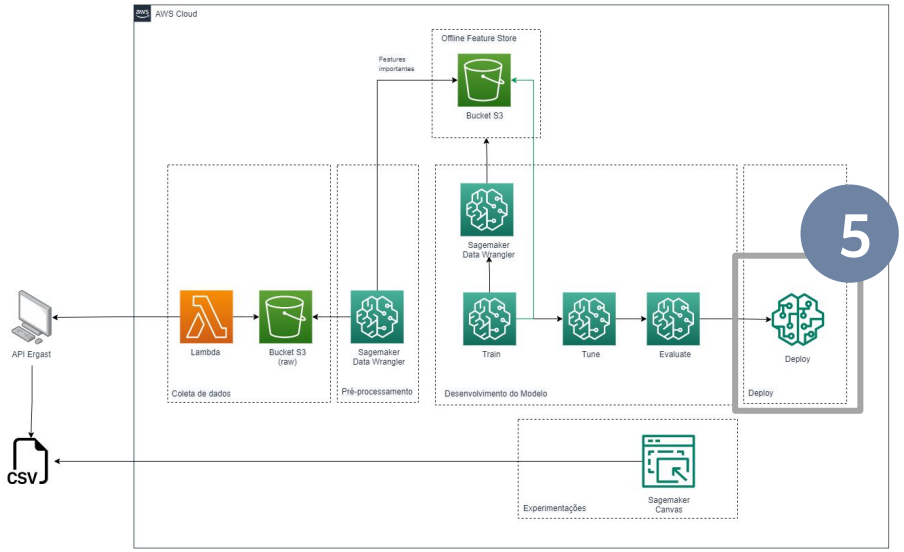
1. Treinamento do modelo
2. Ajuste do modelo
3. **Avaliação do modelo**

DEPLOY DO MODELO



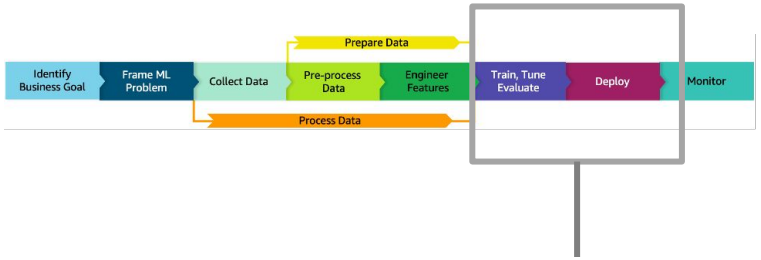
Para o deploy dos modelos, produtizou-se através da função Pickle. Um objeto Pickle permite que possamos utilizá-lo em produção. O código mostrado na sequência foi utilizado em seções anteriores após obtenção de resultados finais.

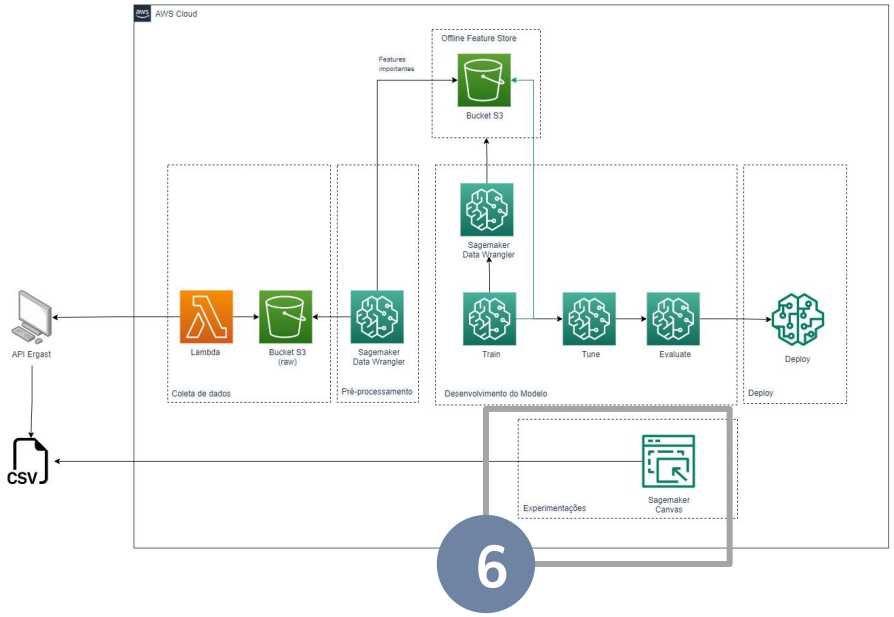
```
import pickle
def save_obj(obj, name):
    with open('models/'+ name, 'wb') as f:
        pickle.dump(obj, f, pickle.HIGHEST_PROTOCOL)

model_filepath = './models/{}.pkl'.format('GradientBoostingClassifier')
print('Saving model ...\n    MODEL: {}'.format(model_filepath))
save_model(rf, model_filepath)
```



1. Deploy com Pickle

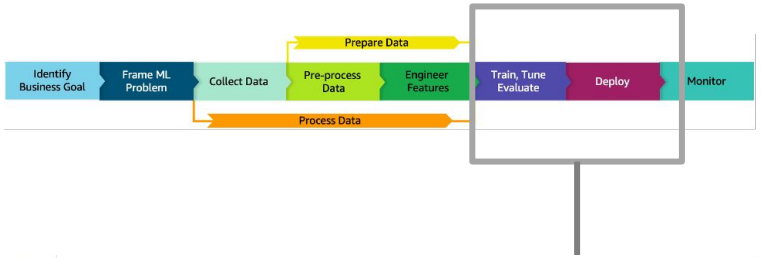
EXPERIMENTAÇÃO NO-CODE DO MODELO

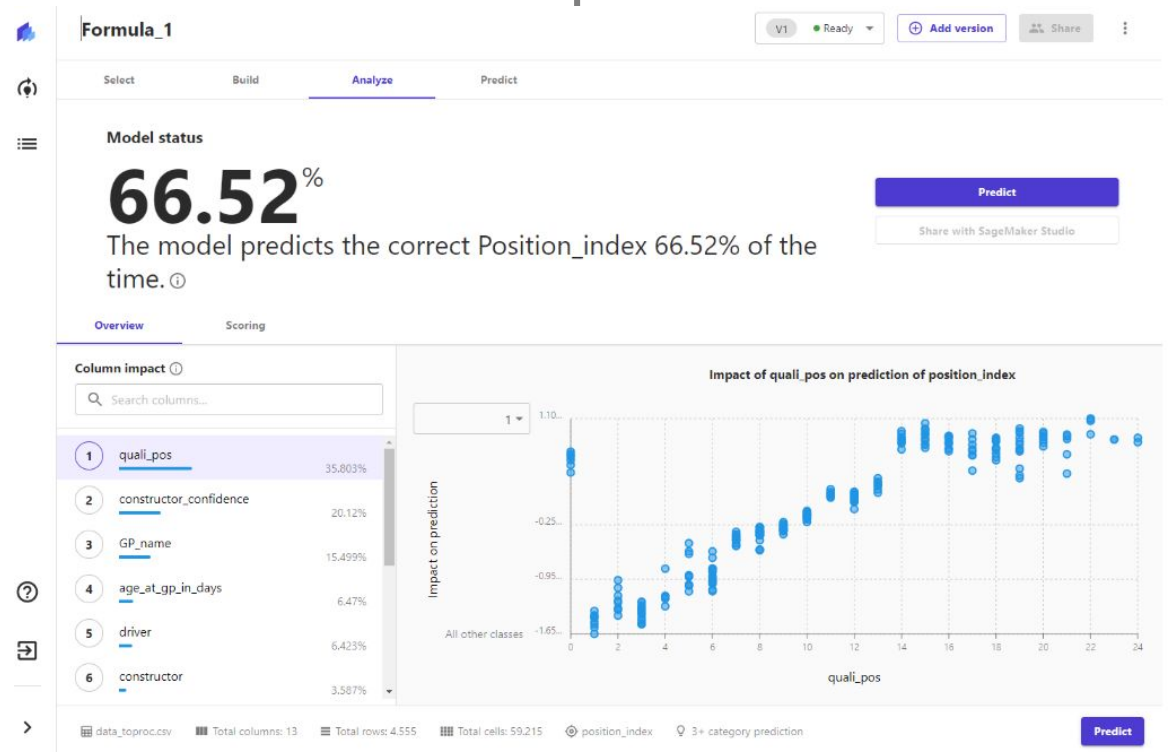


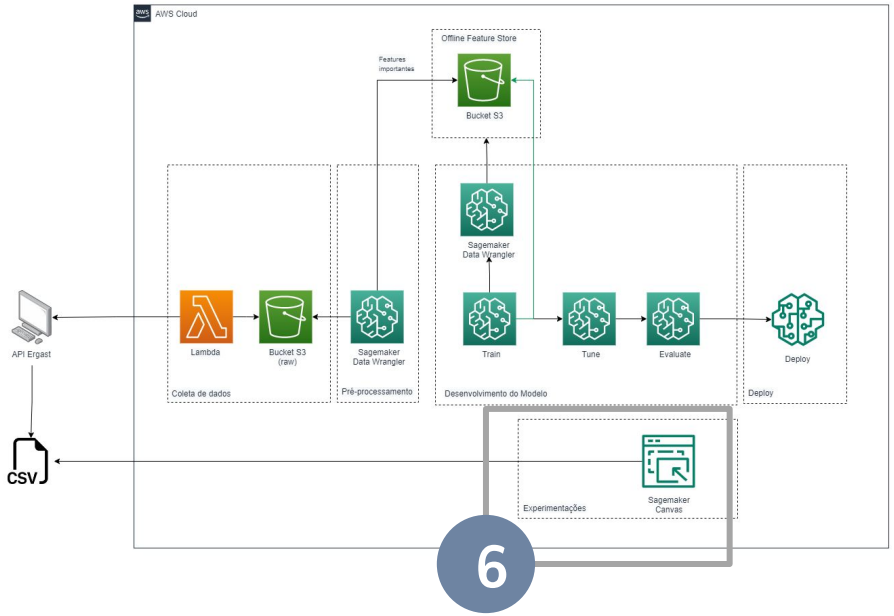


1. Import de datasets (.csv)
2. Predição

EXPERIMENTAÇÃO NO-CODE DO MODELO







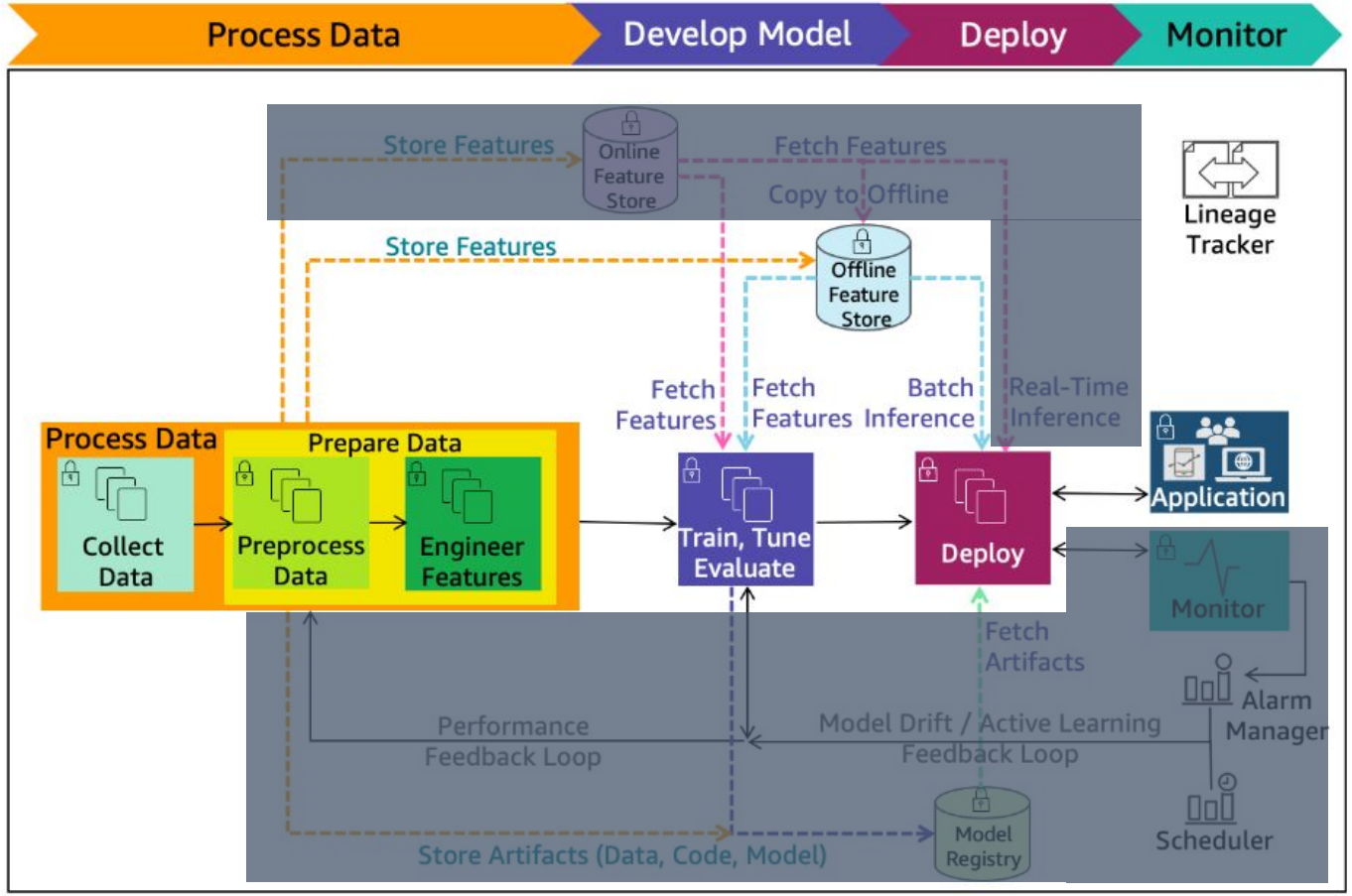
1. Import de datasets (.csv)
2. **Predição**

CONCLUSÕES

...

Aprendizado e aplicações de:

- Frameworks voltado para o mercado
- MLOps: pipelines de entrega contínua e automação no aprendizado de máquina
- Engenharia de dados
- Ciência de dados
- Engenharia de ML



PREDIÇÃO DO CAMPEONATO DE FÓRMULA 1

Mateus Guilherme Fuini
Michelle Melo Cavalcante