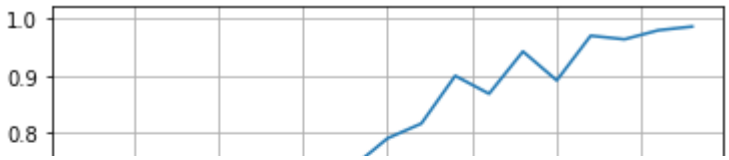
Business Problem + Data Proprocessing

[ ] ↳ *22 cells hidden*

Decision Trees

```
# Hyper-pram tuning + DT model
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score

train_scores = []
val_scores = []

l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)


import matplotlib.pyplot as plt

plt.figure()
plt.plot(list(np.arange(l,u,d)), train_scores, label="train")
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



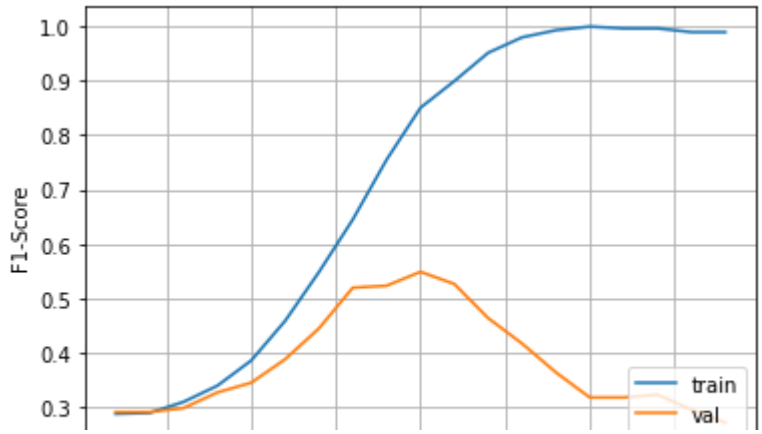

```
# Model with depth_best
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, reca

best_idx = np.argmax(val_scores)
l_best = 2 #l+d*best_idx
print(l_best)
clf = DecisionTreeClassifier(random_state=0, max_depth=l_best, class_weight={ 0:0.1
clf.fit(X_train, y_train)

y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
2
0.41618497109826585
array([[157,  87],
       [ 14,  36]])
```

```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending or
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they mat
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-ax
plt.show() # Show plot
```

Random Forest

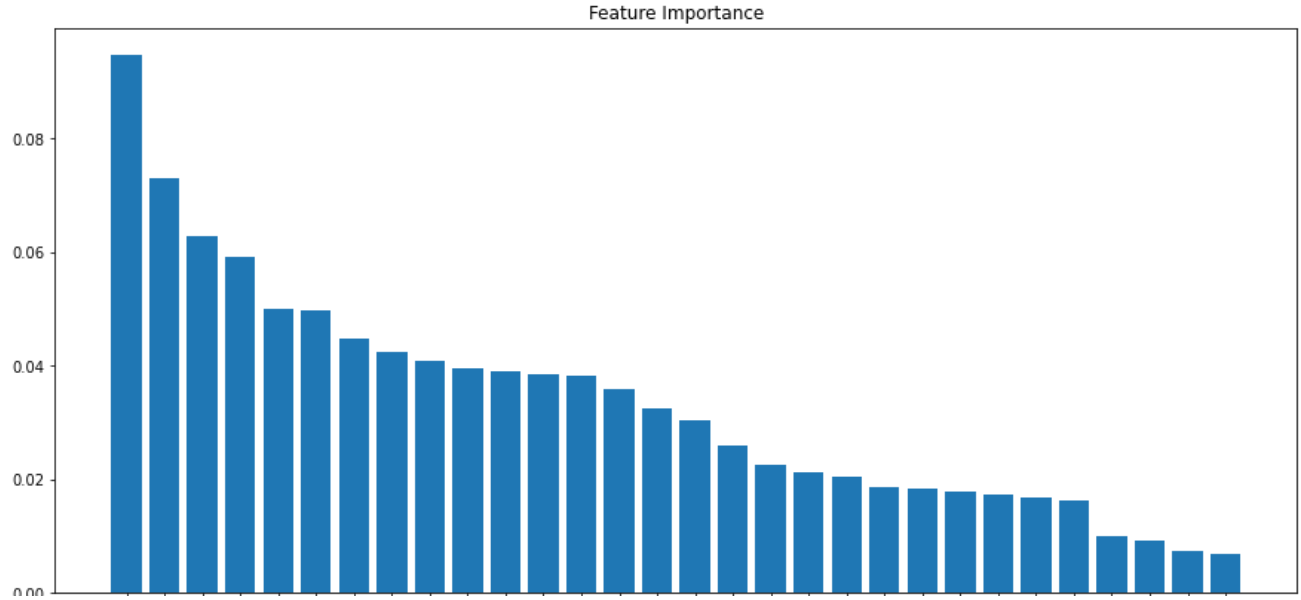

```
# Hyper-pram tuning + DT model
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score

train_scores = []
val_scores = []

l=1
u=20
d=1
w=3.0
num_learners=100
row_sampling_rate = 0.75


for depth in np.arange(l,u,d):
  clf = RandomForestClassifier(max_depth=depth, max_samples=row_sampling_rate, n_es
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)


import matplotlib.pyplot as plt

plt.figure()
plt.plot(list(np.arange(l,u,d)), train_scores, label="train")
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```




```
# Model with depth_best
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, reca

best_idx = np.argmax(val_scores)
l_best = 8 #l+d*best_idx
clf = RandomForestClassifier(max_depth=l_best, max_samples = row_sampling_rate, n_e
clf.fit(X_train, y_train)

y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
0.5194805194805195
array([[180,  64],
       [ 10,  40]])
```

```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending or
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they mat
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-ax
plt.show() # Show plot
```



Other Concepts (covered after boosting)

1. K-Fold CV
2. Grid and Random Search for HPT

✓ 0s completed at 22:27