## Московский государственный технический университет им. Н.Э. Баумана

Факультет Радиотехнический Кафедра РТ5

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4 «Линейные модели, SVM и деревья решений»

Выполнил: Руководитель:

студент группы РТ5-61Б: преподаватель каф. ИУ5

Бабасанова Н. С. Гапанюк Ю.Е.

## Задание

- 1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
- 2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
- 3. С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.
  - 4. Обучите следующие модели:
- о одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
  - o SVM;
  - о дерево решений.
- 5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
- 6. Постройте график, показывающий важность признаков в дереве решений.
- 7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

## Текст программы

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	 worst radius
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	25.380
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	24.990
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	23.570
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	14.910
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	22.540
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	25.450
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	23.690
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	18.980
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	25.740
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	9.456
569 rows × 30 columns											

data\_df["target"] = data.target
data\_df
data\_df.isnull().sum()

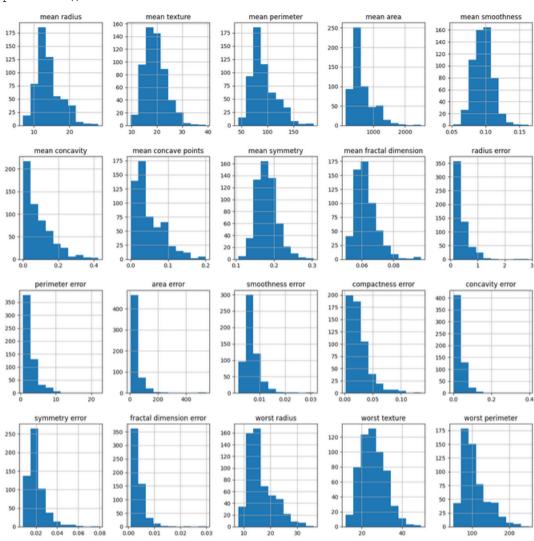
mean radius	0
mean texture	0
mean perimeter	0
mean area	0
mean smoothness	0
mean compactness	0
mean concavity	0
mean concave points	0
mean symmetry	0
mean fractal dimension	0
radius error	0
texture error	0
perimeter error	0
area error	0
smoothness error	0
compactness error	0
concavity error	0
concave points error	0
symmetry error	0
fractal dimension error	0
worst radius	0
worst texture	0
worst perimeter	0
worst area	0
worst smoothness	0
worst compactness	0
worst concavity	0
worst concave points	0
worst symmetry	0
worst fractal dimension	0
target	0
dtype: int64	
· · · · · · · · · · · · · · · · · · ·	

data\_df.describe()

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity		
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000 56		
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799		
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720		
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000		
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560		
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540		
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700		
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800		
8 rows	8 rows × 31 columns								

data\_df.hist(figsize=(20,25))



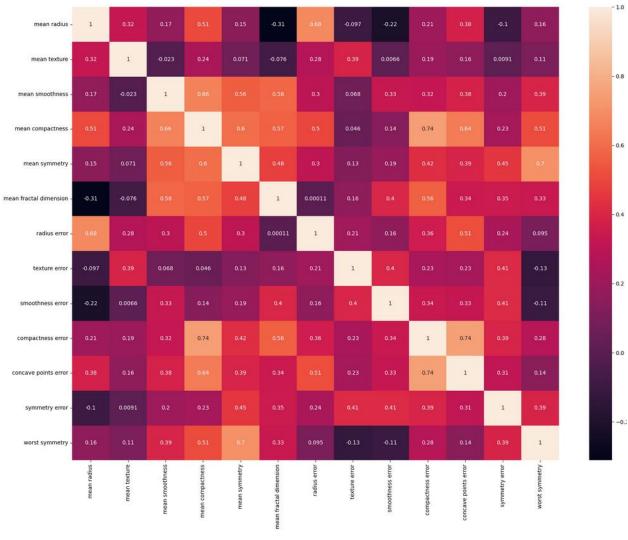


scaler = MinMaxScaler()

```
scaler.fit(data_df.drop('target', axis=1))
scaled_features = scaler.transform(data_df.drop('target',
```

```
axis=1))
```

```
df feat = pd.DataFrame(scaled features,
                                                                                                          columns=data df.columns[:-1])
                      import seaborn as sns
                     plt.figure(figsize=(20,15))
                      ax = sns.heatmap(df feat.corr(),annot=True)
                     plt.show()
              mean radius - 1 0.32 1 0.99 0.17 0.51 0.68 0.62 0.15 0.51 0.68 0.82 0.15 0.51 0.68 0.097 0.67 0.74 0.22 0.21 0.19 0.38 0.1 0.043 0.97 0.3 0.97 0.94 0.12 0.41 0.53 0.74 0.16 0.00
             mean texture - 0.32 1 0.33 0.32 0.023 0.24 0.3 0.29 0.071 0.076 0.28 0.39 0.28 0.26 0.0066 0.19 0.14 0.16 0.00910.054 0.35 0.91 0.36 0.34 0.078 0.28 0.3 0.3 0.3 0.11 0.1:
          mean perimeter - 1 033 1 0.99 0.21 0.56 0.72 0.85 0.18 0.26 0.69 0.087 0.69 0.74 0.2 0.25 0.23 0.41 0.0820.005 0.97 0.3 0.97 0.4 0.15 0.46 0.56 0.77 0.19 0.05
                 mean area - 0.99 0.32 0.99 1 0.18 0.5 0.69 0.82 0.15 0.28 0.73 0.066 0.73 0.8 0.17 0.21 0.21 0.27 0.072 0.02 0.96 0.29 0.96 0.96 0.12 0.39 0.51 0.72 0.14 0.003
       mean smoothness - 0.17 - 0.023 0.21 0.18 1 0.66 0.52 0.55 0.56 0.59 0.3 0.068 0.3 0.25 0.33 0.32 0.25 0.38 0.2 0.28 0.21 0.036 0.24 0.21 0.81 0.47 0.43 0.5 0.39 0.3
                                 0.51 0.24 0.56 0.5 0.66 1 0.88 0.83 0.6 0.57 0.5 0.046 0.55 0.46 0.14 0.74 0.57 0.64 0.23 0.51 0.54 0.25 0.59 0.51 0.5
                                        0.3 0.72 0.69 0.52 0.88 1 0.92 0.5 0.34 0.63 0.076 0.66 0.62 0.099 0.67 0.69 0.68 0.18 0.45 0.69 0.3 0.73 0.68 0.45 0.75 0.88 0.88 0.41 0.5
     mean concave points - 0.82 0.29 0.85 0.82 0.55 0.83 0.92 1 0.46 0.17 0.7 0.021 0.71 0.69 0.028 0.49 0.44 0.62 0.095 0.26 0.83 0.29 0.86 0.81 0.45 0.67 0.75 0.91 0.38 0.3
          mean symmetry - 0.15 0.071 0.18 0.15 0.56 0.6 0.5 0.46 1 0.48 0.3 0.13 0.31 0.22 0.19 0.42 0.34 0.39 0.45 0.33 0.19 0.091 0.22 0.18 0.43 0.47 0.43 0.43 0.47 0.43
mean fractal dimension -0.31-0.076-0.26-0.28 0.58 0.57 0.34 0.17 0.48 1 .000110.16 0.04-0.09 0.4 0.56 0.45 0.34 0.35 0.69 0.25-0.051-0.21-0.23 0.5 0.46 0.35 0.18 0.33 0.77
                                  0.68 0.28 0.69 0.73 0.3 0.5 0.63 0.7 0.3 0.0001 1 0.21 0.97 0.95 0.16 0.36 0.33 0.51 0.24 0.23 0.72 0.19 0.72 0.75 0.14 0.29 0.38 0.53 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.095 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.005 0.00
              texture error -0.097 0.39 0.0870.0660.0680.0460.076 0.021 0.13 0.16 0.21 1 0.22 0.11 0.4 0.23 0.19 0.23 0.41 0.28 0.11 0.41 -0.1 0.0830.0740.0920.069 0.12 0.13 0.04
                                       0.28 0.69 0.73 0.3 0.55 0.66 0.71 0.31 0.04 0.97 0.22 1 0.94 0.15 0.42 0.36 0.56 0.27 0.24 0.7 0.2 0.72 0.73 0.13 0.34 0.42 0.55 0.11 0.085
                                 0.74 0.26 0.74 0.8 0.25 0.46 0.62 0.69 0.22 0.09 0.95 0.11 0.94 1 0.075 0.28 0.27 0.42 0.13 0.13 0.76 0.2 0.76 0.81 0.13 0.28 0.39 0.54 0.0740.018
        smoothness error - 0.220,0066 0.2 0.17 0.33 0.14 0.099 0.028 0.19 0.4 0.16 0.4 0.15 0.075 1 0.34 0.27 0.33 0.41 0.43 0.23 0.075 0.22 0.18 0.31 0.0560.058 0.1 0.11 0.1
      compactness error - 0.21 0.19 0.25 0.21 0.32 0.74 0.67 0.49 0.42 0.56 0.36 0.23 0.42 0.28 0.34 1 0.8 0.74 0.39 0.8 0.2 0.14 0.26 0.2 0.23 0.68 0.
           concavity error - 0.19 0.14 0.23 0.21 0.25 0.57 0.69 0.44 0.34 0.45 0.33 0.19 0.36 0.27 0.27 0.8 1 0.77 0.31 0.73 0.19 0.1 0.23 0.19 0.1 0.48 0.66
    concave points error - 0.38 0.16 0.41 0.37 0.38 0.64 0.68 0.62 0.39 0.34 0.51 0.23 0.56 0.42 0.33 0.74 0.77 1 0.31 0.61 0.36 0.087 0.39 0.34 0.22 0.45 0.55 0.6 0.14 0.31
          symmetry error - 0.1 0.00910.0820.072 0.2 0.23 0.18 0.095 0.45 0.35 0.24 0.41 0.27 0.13 0.41 0.39 0.31 0.31 1 0.37 0.13 0.077 0.1 0.11 0.013 0.06 0.037 0.03 0.39 0.078
 fractal dimension error -0.0430.0540.00550.02 0.28 0.51 0.45 0.26 0.33 0.69 0.23 0.28 0.24 0.13 0.43 0.8 0.73 0.61 0.37 1 0.0370.00320.0010.023 0.17 0.39 0.38 0.22 0.11 0.5
               worst radius - 0.97 0.35 0.97 0.96 0.21 0.54 0.69 0.83 0.19 0.25 0.72 0.11 0.7 0.76 0.23 0.2 0.19 0.36 0.13 0.037 1 0.36 0.99 0.98 0.22 0.48 0.57 0.79 0.24 0.093
             worst texture - 0.3 0.91 0.3 0.29 0.036 0.25 0.3 0.29 0.091 0.051 0.19 0.41 0.2 0.2 0.075 0.14 0.1 0.087 0.0770.0032036 1 0.37 0.35 0.23 0.36 0.37 0.36 0.23 0.22
          worst perimeter - 0.97 0.36 0.97 0.96 0.24 0.59 0.73 0.86 0.22 -0.21 0.72 -0.1 0.72 -0.1 0.72 0.76 -0.22 0.26 0.23 0.39 -0.1 -0.001 0.99 0.37 1 0.98 0.24 0.53 0.62 0.82 0.27 0.14
                 worst area - 0.94 0.34 0.94 0.96 0.21 0.51 0.68 0.81 0.18 0.23 0.75 0.083 0.73 0.81 0.18 0.22 0.75 0.083 0.73 0.81 0.18 0.2 0.19 0.34 0.11 0.023 0.98 0.35 0.98 1 0.21 0.44 0.54 0.75 0.21 0.08
       worst smoothness - 0.12 0.078 0.15 0.12 0.81 0.57 0.45 0.45 0.45 0.43 0.5 0.14 0.074 0.13 0.13 0.31 0.23 0.17 0.22 0.013 0.17 0.22 0.23 0.24 0.21 1 0.57 0.52 0.55 0
      worst compactness - 0.41 0.28 0.46 0.39 0.47 0.87 0.75 0.67 0.47 0.46 0.29 0.092 0.34 0.28 0.056 0.68 0.48 0.45 0.06 0.39 0.48 0.36 0.53 0.44 0.5
          worst concavity - 0.53 0.3 0.56 0.51 0.43 0.82 0.88 0.75 0.43 0.35 0.38 -0.069 0.42 0.39 0.058 0.64 0.66 0.55 0.037 0.38 0.57 0.37 0.62 0.54 0.52 0.89 1 0.86 0.53 0.0
    worst concave points - 0.74 0.3 0.77 0.72 0.5 0.82 0.86 0.91 0.43 0.18 0.53 -0.12 0.55 0.54 0.1 0.48 0.44 0.6 0.03 0.22 0.79 0.36 0.82 0.75 0.55 0.8 0.86 1 0.5 0.5
          worst symmetry - 0.16 0.11 0.19 0.14 0.39 0.51 0.41 0.38 0.7 0.33 0.095 0.13 0.11 0.074 0.11 0.28 0.2 0.14 0.39 0.11 0.24 0.23 0.27 0.21 0.49 0.61 0.53 0.5 1 0.54
worst fractal dimension -0.0071 0.12 0.0510.0037 0.5 0.69 0.51 0.37 0.44 0.77 0.05-0.0460.085 0.018 0.1 0.59 0.44 0.31 0.078 0.59 0.093 0.22 0.14 0.08 0.62 0.81 0.69 0.51 0.54 1
                                  ilius - Tror - T
                         corr matrix = df feat.corr().abs()
                      # Верхний треугольник матрицы, чтобы не учитывать дубли
                                                                                            corr matrix.where(np.triu(np.ones(corr matrix.shape),
                      upper
k=1).astype(bool))
                      to drop = [column for column in upper.columns if any(upper[column] > 0.8)]
                     df reduced = df feat.drop(columns=to drop)
                     df reduced
plt.figure(figsize=(20,15))
ax = sns.heatmap(df reduced.corr(),annot=True)
plt.show()
```



```
X = df reduced
y = data_df["target"]
from sklearn.model selection import train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random state=42)
# ----- Логрег
from sklearn.linear model import LogisticRegression
logreg = LogisticRegression(random state=42)
logreg.fit(X train, y train)
y_pred_logreg = logreg.predict(X_test)
# ---- SVM
from sklearn.svm import LinearSVC, SVC
# X = X.iloc[:, :2]
# svm = LinearSVC(C=1.0, loss='hinge')
svm = SVC(kernel='rbf', C=1.0, gamma='scale')
svm.fit(X, y)
y pred svm = svm.predict(X test)
```

```
# ----- Дерево решений
from sklearn.tree import DecisionTreeClassifier, plot_tree

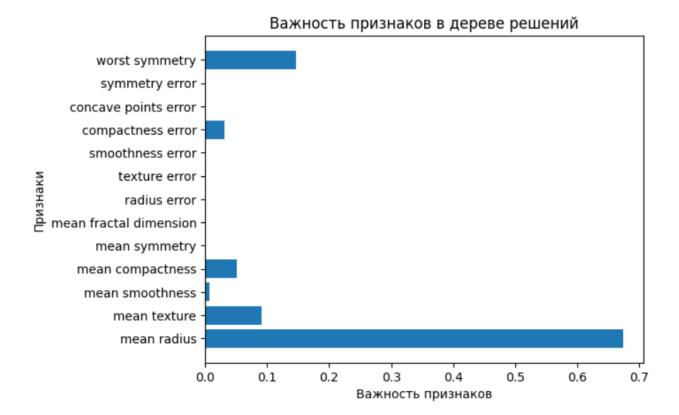
tree = DecisionTreeClassifier(max_depth=3, random_state=42)
tree.fit(X_train, y_train)
y_pred_tree = tree.predict(X_test)

from sklearn.metrics import accuracy_score, classification_report
# print("Jorper accuracy:", accuracy_score(y_test, y_pred_logreg))
# print("SVM accuracy:", accuracy_score(y_test, y_pred_svm))
# print("Дерево решений accuracy:", accuracy_score(y_test, y_pred_tree))

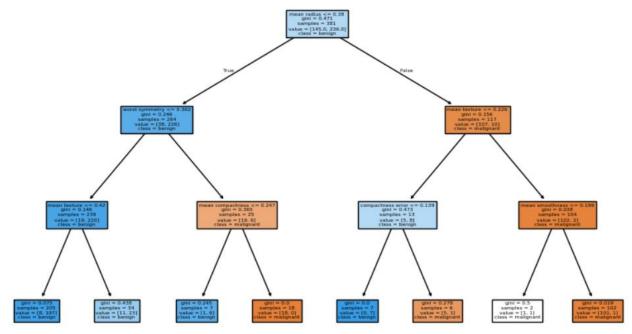
print("\nЛогрег репорт\n", classification_report(y_test, y_pred_logreg, digits=4))
print("\nSVM репорт\n", classification_report(y_test, y_pred_svm, digits=4))
print("\nДерево решений репорт\n", classification_report(y_test, y_pred_svm, digits=4))
print("\nДерево решений репорт\n", classification_report(y_test, y_pred_tree, digits=4))
```

Логрег репорт				
	precision	recall	f1-score	support
9	0.9516	0.8806	0.9147	67
_				
1	0.9365	0.9752	0.9555	121
accuracy			0.9415	188
macro avg	0.9441	0.9279	0.9351	188
weighted avg	0.9419	0.9415	0.9409	188
weighted avg	0.5415	0.3413	0.3403	100
SVM репорт				
	precision	recall	f1-score	support
0	0.9552	0.9552	0.9552	67
1	0.9752	0.9752	0.9752	121
accuracy			0.9681	188
macro avg	0.9652	0.9652	0.9652	188
weighted avg	0.9681	0.9681	0.9681	188
5 5				
Дерево решений	репорт			
	precision	recall	f1-score	support
0	0.8971	0.9104	0.9037	67
1	0.9500	0.9421	0.9461	121
accuracy			0.9309	188
macro avg	0.9235	0.9263	0.9249	188
weighted avg	0.9311	0.9309	0.9310	188

```
# Важность признаков в дереве решений feature_importances = tree.feature_importances_ plt.barh(X.columns, feature_importances) plt.xlabel("Важность признаков") plt.ylabel("Признаки") plt.title("Важность признаков в дереве решений") plt.show()
```



# Визуализация дерева решений plt.figure(figsize=(10, 6)) plot\_tree(tree, feature\_names=X.columns, class\_names=["malignant", "benign"], filled=True) plt.show()



# Правила дерева решений в текстовом виде from sklearn.tree import export\_text tree rules = export text(tree, feature names=list(X.columns))

print("Правила решений дерева в текстовом виде:")
print(tree\_rules)

```
Правила решений дерева в текстовом виде:
 --- mean radius <= 0.38
    --- worst symmetry <= 0.36
       |--- mean texture <= 0.42
       | |--- class: 1
       |--- mean texture > 0.42
       | |--- class: 1
    |--- worst symmetry > 0.36
       |--- mean compactness <= 0.25
       | |--- class: 1
        |--- mean compactness > 0.25
       | |--- class: 0
 --- mean radius > 0.38
    --- mean texture <= 0.23
       --- compactness error <= 0.14
           --- class: 1
       |--- compactness error > 0.14
       | |--- class: 0
    |--- mean texture > 0.23
       |--- mean smoothness <= 0.20
            |--- class: 0
        --- mean smoothness > 0.20
           |--- class: 0
```