Tutorium 1

Table of contents

- Tutorium 1
  - Table of contents
  - Task 1: Decision Trees in Python
  - Task 2: Automatic Decision Trees

Task 1: Decision Trees in Python

The data includes the **characteristics of each Titanic passenger** (e.g., age, class) and whether or not they **survived the tragedy**.

1. Familiarise yourself with the data set. Count how many people embarked from each city using the .value_counts() command.

```
df.value_counts("embarked")
```

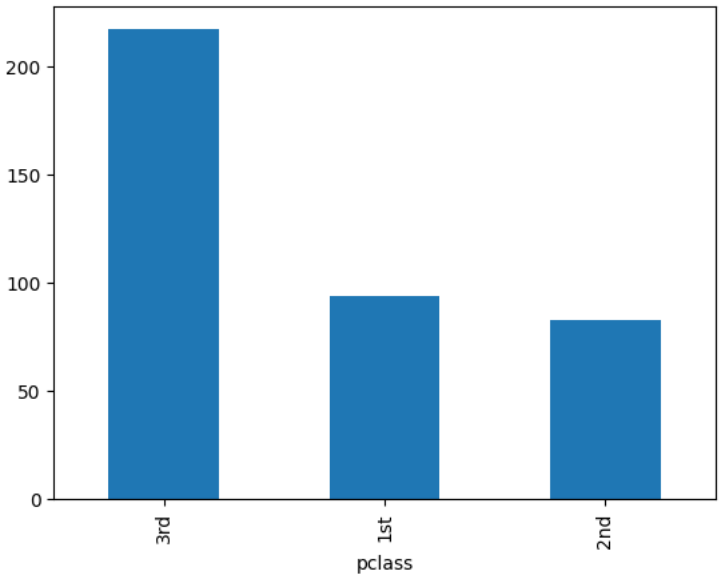
Output:

```
embarked
Southampton    174
Cherbourg       58
Queenstown      18
Name: count, dtype: int64
```

2. Create a barplot that shows how many people were in each class (pclass). Include the bar chart in your pdf.

```
df['pclass'].value_counts(ascending=False).plot(kind='bar')
```

Output:



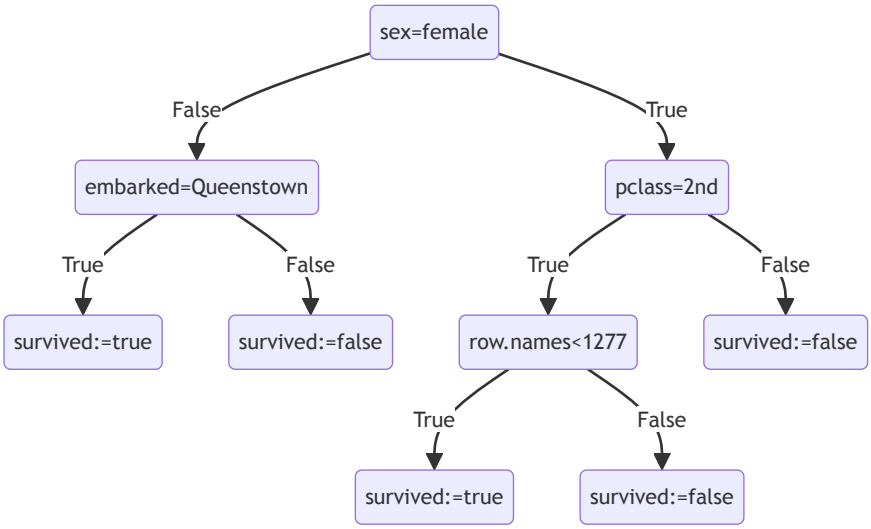
3. Select the first 15 rows and save them as a data frame named df_subset. Copy the resulting table in your submission pdf.

```
df_subset = df.head(15).copy()
```

Output:

	row.names	pclass	survived	name	age	embarked	home.dest	room	ticket	boat	sex
703	704	3rd	0	Cann, Mr Ernest	21.0	Southampton	NaN	NaN	NaN	NaN	male
260	261	1st	1	Taylor, Mrs Elmer Zebley (Juliet Cummins Wright)	NaN	Southampton	London / East Orange, NJ	C-126	NaN	5	female
153	154	1st	0	Kent, Mr Edward Austin	58.0	Cherbourg	Buffalo, NY	NaN	NaN	(258)	male
606	607	3rd	1	Abelseth, Miss Anna Karen	16.0	Southampton	Norway Los Angeles, CA	NaN	NaN	16	female
448	449	2nd	0	Hold, Mr Stephen	42.0	Southampton	England / Sacramento, CA	NaN	NaN	NaN	male
440	441	2nd	0	Hickman, Mr Lewis	32.0	Southampton	West Hampstead, London / Neepawa, MB	NaN	NaN	NaN	male
517	518	2nd	0	Norman, Mr Robert Douglas	NaN	Southampton	Glasgow	NaN	NaN	(287)	male
933	934	3rd	1	Kink, Miss Louise Gretchen	NaN	NaN	NaN	NaN	NaN	NaN	female
1276	1277	3rd	0	Van Impe, Miss Catharine	NaN	NaN	NaN	NaN	NaN	NaN	female
487	488	2nd	0	Mack, Mrs Mary	57.0	Southampton	Southampton / New York, NY	E77	NaN	(52)	female
692	693	3rd	1	Buckley, Mr Daniel	21.0	Queenstown	Kingwilliamstown, Co Cork, Ireland New York, NY	NaN	NaN	NaN	male
186	187	1st	0	Natsch, Mr Charles H.	36.0	Cherbourg	Brooklyn, NY	NaN	NaN	NaN	male
899	900	3rd	0	Johnston, Mrs Andrew G.	NaN	NaN	NaN	NaN	NaN	NaN	female
576	577	2nd	0	Ware, Mr William J.	23.0	Southampton	NaN	NaN	NaN	NaN	male
6	7	1st	1	Andrews, Miss Kornelia Theodosia	63.0	Southampton	Hudson, NY	D-7	13502 L77	10	female

4. Draw a decision tree for df_subset using pclass, age, embarked, home.dest, and sex as features, and survived as the target variable.



5. Next, implement your decision tree in Python and apply the decision tree to the complete data set.

```
def survival_decision_tree(row):
    if (row["sex"] == "female"\
            and row["pclass"] != "2nd"\
            and row["row.names"] < 1277)\
            or (row["sex"] == "male"\
            and row["embarked"] == "Queenstown"):
        return True
    else:
        return False

# apply decision tree function to subset
df_subset['predicted_survival'] = df_subset.apply(survival_decision_tree, axis=1)
df_subset[df_subset["predicted_survival"] == True]
```

Output:

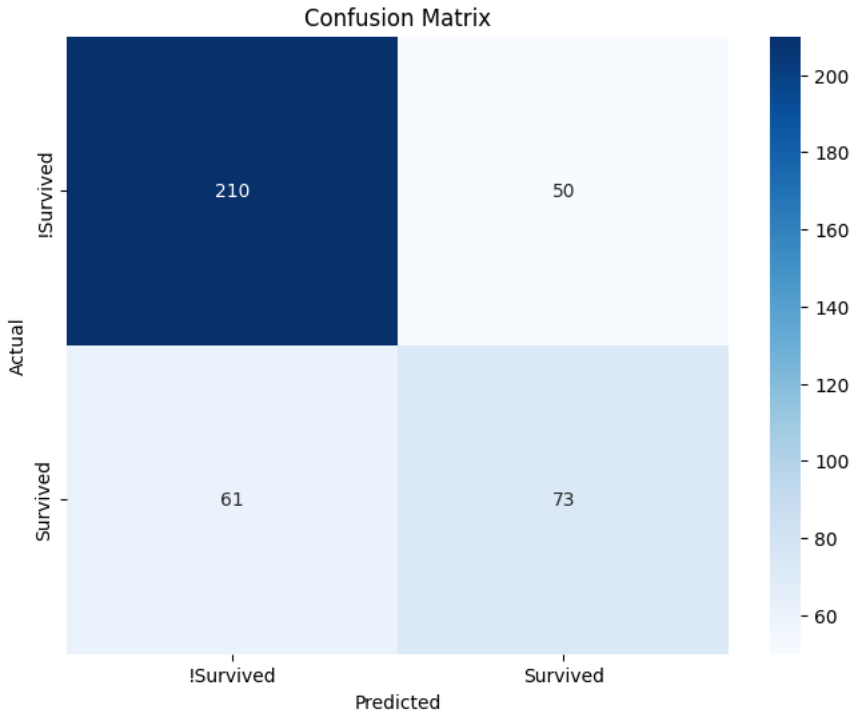
	row.names	pclass	survived	name	age	embarked	home.dest	room	ticket	boat	sex	predic
260	261	1st	1	Taylor, Mrs Elmer Zebley (Juliet Cummins Wright)	NaN	Southampton	London / East Orange, NJ	C-126	NaN	5	female	True
606	607	3rd	1	Abelseth, Miss Anna Karen	16.0	Southampton	Norway Los Angeles, CA	NaN	NaN	16	female	True
933	934	3rd	1	Kink, Miss Louise Gretchen	NaN	NaN	NaN	NaN	NaN	NaN	female	True
692	693	3rd	1	Buckley, Mr Daniel	21.0	Queenstown	Kingwilliamstown, Co Cork, Ireland New York, NY	NaN	NaN	NaN	male	True
899	900	3rd	0	Johnston, Mrs Andrew G.	NaN	NaN	NaN	NaN	NaN	NaN	female	True

	row.names	pclass	survived	name	age	embarked	home.dest	room	ticket	boat	sex	predic
6	7	1st	1	Andrews, Miss Kornelia Theodosia	63.0	Southampton	Hudson, NY	D-7	13502 L77	10	female	True

6. Generate a confusion matrix to compare the predictions based on your decision tree with the actual survival data. A confusion matrix shows the number of true positives, true negatives, false positives, and false negatives. You may read more about what a confusion matrix is here: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. Please use sklearn to implement the confusion matrix. The following page shows an example: https://scikit-learn.og/stable/modules/generated/sklearn.metrics.confusion_matrix.html

```
actual = df["survived"]
predicted = df["predicted_survival"]
plt.figure(figsize=(8, 6))
sb.heatmap(cm(actual, predicted),
            annot=True,
            fmt="d",
            cmap="Blues",
            xticklabels=["!Survived", "Survived"],
            yticklabels=["!Survived", "Survived"])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Output:
Here's the confusion matrix of our manually created decision tree:



7. Why might your decision tree not correctly classify all new data points? Is this a problem of over- or under-fitting?

Limited data and complex relationships as well as overfitting are reason that might lead to the poor result of our heuristic approach.

Is this a problem of over- or under-fitting?
Our decision tree is clearly overfitting as 101 results were misclassified as the on the whole dataset (df) but only 1 in the original training batch (df_subset).

- One example of overfitting is that the condition accurately represent the training data (df_subset) but does not work accurately on the whole dataset (df).

- One example of underfitting would be that the condition does not accurately represent the training data (df_subset) and would perform approximately the same way on the whole dataset (df).

8. Neural networks often achieve better accuracy than machine learning approaches based on decision trees, such as SVMs. Nonetheless, decision-tree-based machine learning approaches are most widely used. What do you think are the advantages of these approaches? In particular, think about the creditworthiness data set we used in the tutorial.

Decision trees such as support vectors machine (SVMs), offer interpretability, efficiency, robustness, and ease of use for creditworthiness analysis, but may lack neural networks' accuracy and efficiency. Also, SVMs are pretty computationnaly expensive compared to good neural network approaches.

Task 2: Automatic Decision Trees

You can use the library sklearn to calculate a decision tree automatically. Calculate an automatic decision tree for the Titanic data set and visualize the tree using graphviz and pydotplus.

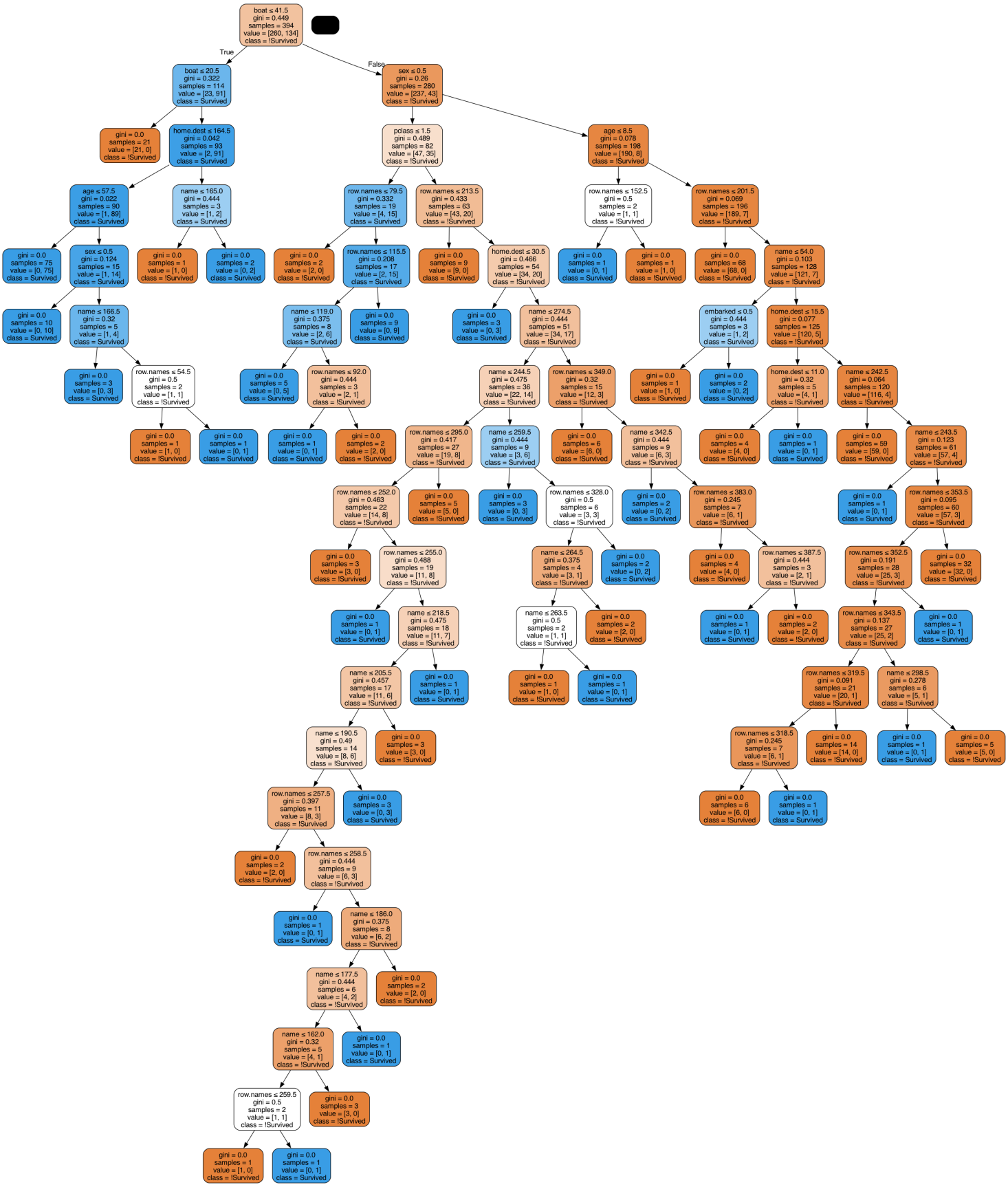
```
le = LabelEncoder()
df_copy = df_copy.apply(le.fit_transform)

# X -> predictions
# y -> labels (targets)
X = df_copy.drop(columns=["survived"])
y = df_copy["survived"]

clf = DecisionTreeClassifier()
clf.fit(X, y)

dot_data = export_graphviz(
    clf, out_file=None,
    feature_names=X.columns,
    class_names=['!Survived',
    'Survived'],
    filled=True,
    rounded=True,
    special_characters=True
    )
graph = pydotplus.graph_from_dot_data(dot_data)
graph.write_png('mydecisiontree.png')
```

Output:



Just for testing purposes:

This time we regenerated the cm but with the same training and validation dataset. As we can see the Decision Tree Classifier clearly **overfit** our data as this one would not perform the same way on new unseen data. It has a 100% accuracy because the decision tree was created to completely use all the features available for classifying the outcome.

```
# Just for testing purposes

new_predictions = clf.predict(X)

plt.figure(figsize=(8, 6))
sb.heatmap(cm(y, new_predictions),
            annot=True,
            fmt="d",
            cmap="Blues",
            xticklabels=["!Survived", "Survived"],
            yticklabels=["!Survived", "Survived"])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Output: