Adult Income Predictor Report

- DSCI 522 - Workflows
- MDS 2024-2025
- Group 24
- Members: Michael Suriawan, Francisco Ramirez, Tingting Chen, Quanhua Huang

Summary

This report presents the application of a K-Nearest Neighbors (KNN) Classifier to predict an individual's annual income based on selected categorical socioeconomic features from the Adult dataset. The dataset, sourced from the 1994 U.S. Census Bureau, contains 48,842 instances and features such as age, education, occupation, and marital status. The model achieved an accuracy of approximately 80%, with a tendency to predict more individuals with incomes below $50K compared to those above. This result emphasizes the importance of socioeconomic factors in determining income levels. Further investigation into individual feature contributions and the inclusion of numerical variables like age and hours-per-week could enhance prediction performance.

Introduction

The Adult dataset, originally curated from the 1994 U.S. Census Bureau database, is a well-known benchmark dataset in machine learning. Its primary objective is to predict whether an individual earns more or less than $50,000 annually based on various demographic and socio-economic attributes. With 48,842 instances and 14 features, the dataset encompasses a mix of categorical and continuous variables, making it a rich resource for classification tasks and exploratory data analysis.

The model described in this notebook, looks to use a trained "Nearest Neighbors" Classifier to use different socioeconomic features to predict the range of the individual's income. The features in the data set include characteristics such as age, education level, marital status, occupation, among others.

The model looks to predict whether or not an individual's income exceeds $50K/yr based on the selected categorical socioeconomic features. For simplicity, only selected categorical features from the original data set. These features are specifically encoded based on their content prior to training the kNN classifier used for predictions.

Note: The original data set's reference information can be found at the end of this document.

Setup

In [1]:
```
import os
import requests
import zipfile
import pandas as pd
import altair as alt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import ConfusionMatrixDisplay
alt.data_transformers.enable('vegafusion')
```

Out[1]:
```
DataTransformerRegistry.enable('vegafusion')
```

Data Load

The following cell loads and displays the original data set.

It also adds names to the columns aligned to the description from the data set location in the UC Irvine Machine Learning Repository.

In [2]:
```
# download data as zip and extract
url = "https://archive.ics.uci.edu/static/public/2/adult.zip"

request = requests.get(url)
os.makedirs("../data/raw", exist_ok=True)

with open("../data/raw/adult.zip", 'wb') as f:
    f.write(request.content)

with zipfile.ZipFile("../data/raw/adult.zip", 'r') as zip_ref:
    zip_ref.extractall("../data/raw")

data_adult = pd.read_csv("../data/raw/adult.data", names = ['age', 'workclas
data_adult.head()
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife

EDA Analysis

Data Summary

Below is the summary of our dataset which contains numerical and categorical variables

In [3]:
```
data_adult.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64 
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64 
 11  capital-loss    32561 non-null  int64 
 12  hours-per-week  32561 non-null  int64 
 13  native-country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [4]:
```
data_adult.describe()
```

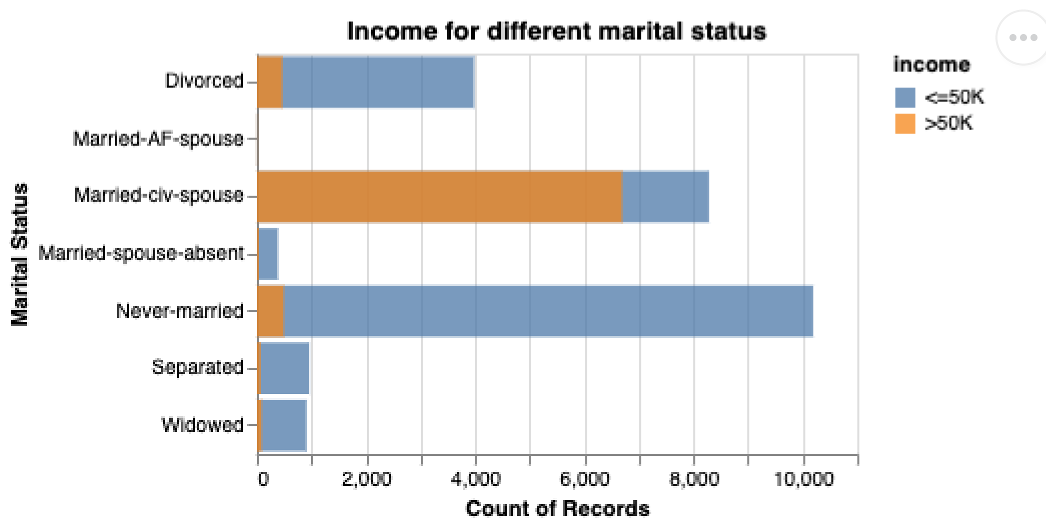
Out[4]:

	age	fnlwgt	education-num	capital-gain	capital-loss	h
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	3256
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	4
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	4
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	4
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	4
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	9

Visualization of Dataset

In [5]:
```
alt.Chart(data_adult, title = "Income for different marital status").mark_ba
    alt.Y('marital-status').title("Marital Status"),
    alt.X('count()').stack(False),
    alt.Color('income')
).properties(
    height=200,
    width=300
)
```

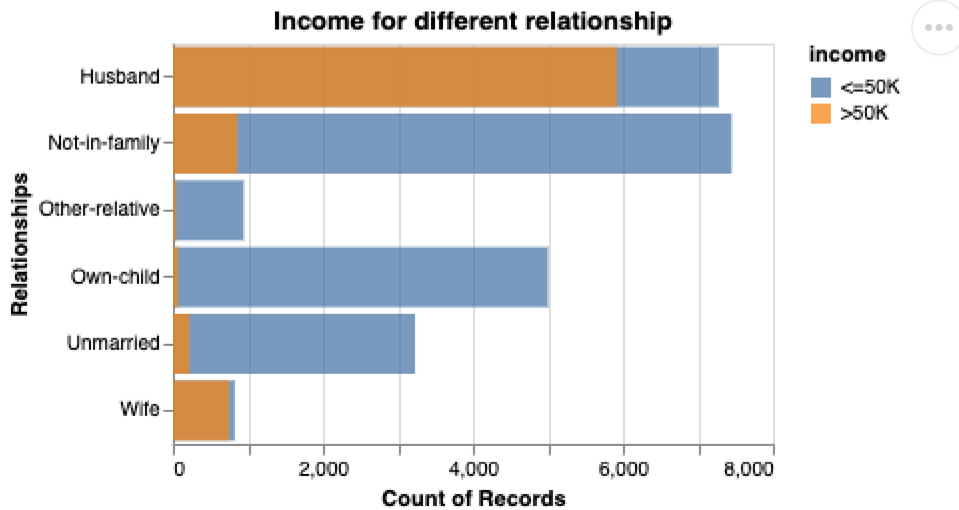
Out[5]:



In [6]:
```
alt.Chart(data_adult, title = "Income for different relationship").mark_bar(
    alt.Y('relationship').title("Relationships"),
    alt.X('count()').stack(False),
    alt.Color('income')
).properties(
    height=200,
```

```
    width=300
)
```

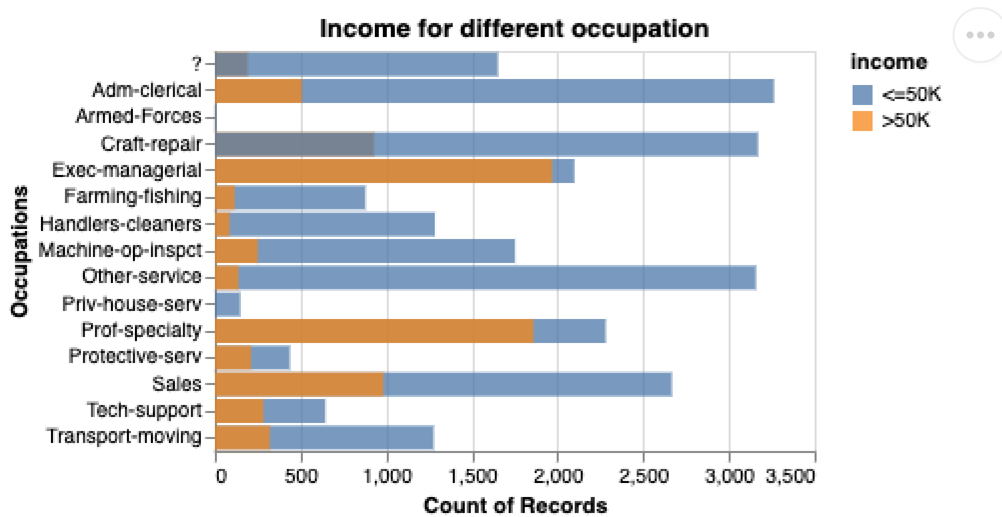
Out[6]:



In [7]:

```
alt.Chart(data_adult, title = "Income for different occupation").mark_bar(op
    alt.Y('occupation').title("Occupations"),
    alt.X('count()').stack(False),
    alt.Color('income')
).properties(
    height=200,
    width=300
)
```

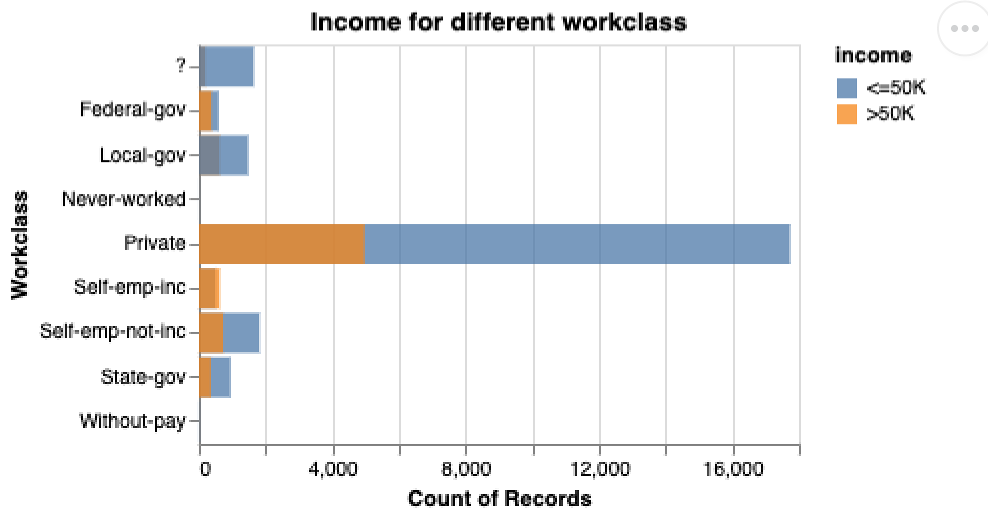
Out[7]:



In [8]:

```
alt.Chart(data_adult, title = "Income for different workclass").mark_bar(opa
    alt.Y('workclass').title("Workclass"),
    alt.X('count()').stack(False),
    alt.Color('income')
).properties(
    height=200,
    width=300
)
```

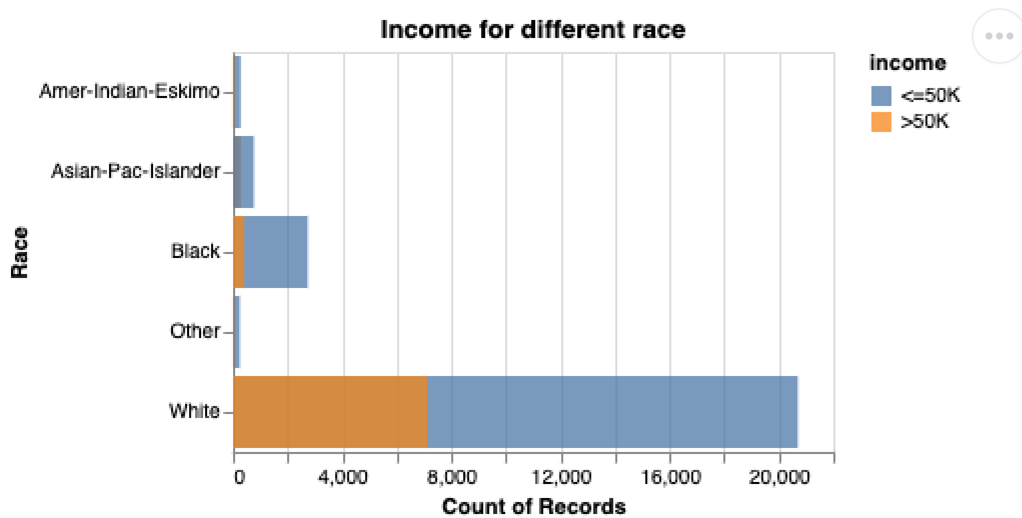
Out[8]:



In [9]:

```python
alt.Chart(data_adult, title = "Income for different race").mark_bar(opacity
    alt.Y('race').title("Race"),
    alt.X('count()').stack(False),
    alt.Color('income')
).properties(
    height=200,
    width=300
)
```

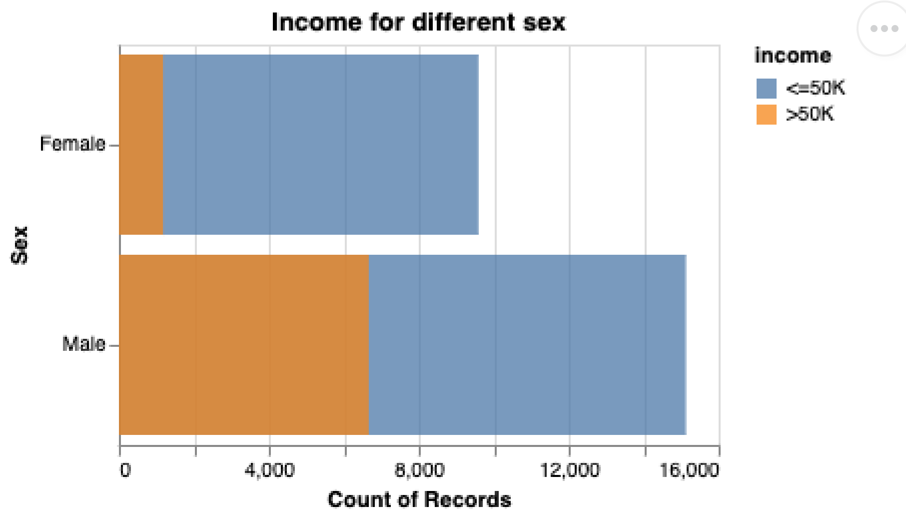
Out[9]:



In [10]:

```python
alt.Chart(data_adult, title = "Income for different sex").mark_bar(opacity =
    alt.Y('sex').title("Sex"),
    alt.X('count()').stack(False),
    alt.Color('income')
).properties(
    height=200,
    width=300
)
```

Out[10]:



Train/Test Split

The following cell separates the data set into train and test sets for purposes of training the classifier model.

It uses an 80/20 data split for training and test.

It also defines the target colums, which will be the income range (Column = income)

In [11]:
```
train_df, test_df = train_test_split(data_adult, test_size=0.20, random_stat
X_train, y_train = (
    train_df.drop(columns=['income']),
    train_df["income"],
)
X_test, y_test = (
    test_df.drop(columns=['income']),
    test_df["income"],
)
```

Column Selection

The following cell describes which columns were selected to train the classifier model.

For simplicity, the model is focused on using categorical variables available in the data set.

In [12]:
```
categorical_features = ["marital-status", "relationship", "occupation", "wor
binary_features = ["sex"]
drop_features = ["age", "fnlwgt", "education", "education-num", "capital-gai
```

Preprocessing

The following cell uses One Hot Encoder to encode categorical features, as well as using a Simple Imputer to deal with missing data in the data set.

Additionally, it creates a Column Transformer describing the treatment that each column will get during the encoding process.

In [13]:

```
binary_transformer = OneHotEncoder(drop="if_binary", dtype=int)

categorical_transformer = make_pipeline(
    SimpleImputer(strategy="constant", fill_value="missing"),
    OneHotEncoder(handle_unknown="ignore", sparse_output=False),
)

preprocessor = make_column_transformer(
    (binary_transformer, binary_features),
    (categorical_transformer, categorical_features),
    ("drop", drop_features),
)
```

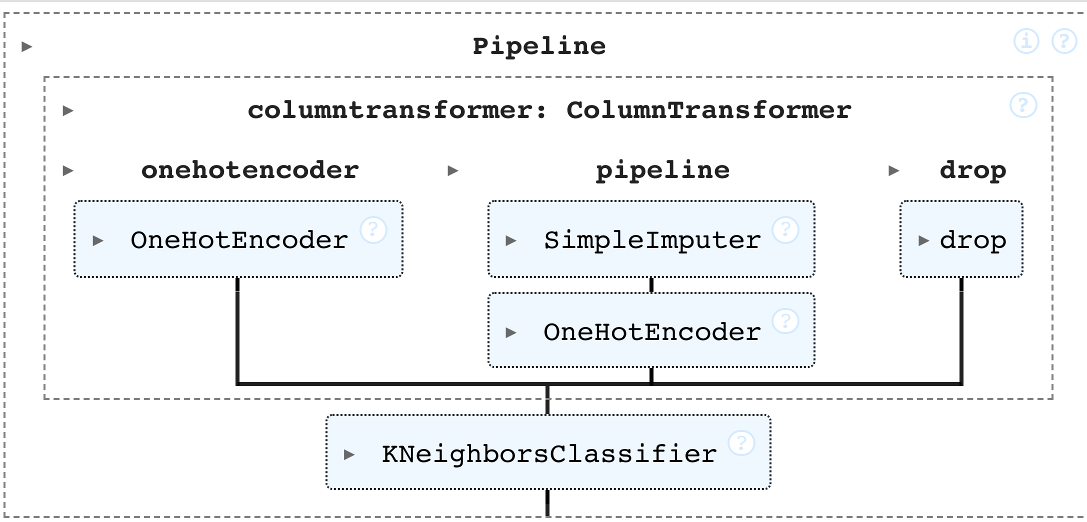
Model Fit

A pipeline is created that describes the preprocessing and KNN flow that will be used to train the model with "fit". Immediately after, the model's performance score is displayed based on training data.

In [14]:

```
model = KNeighborsClassifier()
pipe = make_pipeline(preprocessor, model)
pipe.fit(X_train, y_train)
```

Out[14]:



In [15]:

```
pipe.score(X_train, y_train)
```

Out[15]: 0.8179130835380836

Model Test Score and Prediction

Finally, the model is scored on the unseen examples.

Additionally, it displays the hard predictions the model does on the test data.

In [16]:
```
test_score = pipe.score(X_test, y_test)
test_score
```

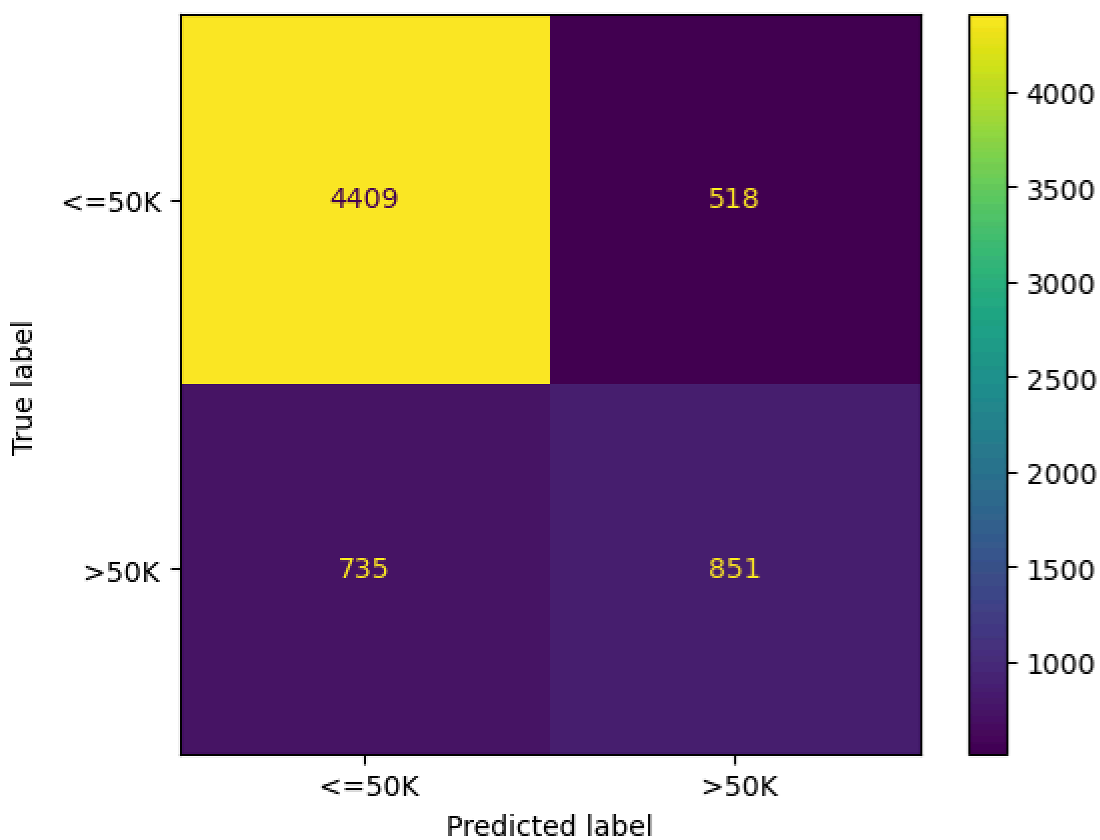
Out[16]: 0.8076155381544603

Visualization of the prediction result

In [17]:
```
cm = ConfusionMatrixDisplay.from_estimator(
    pipe,
    X_test,
    y_test,
    values_format="d"
)

cm
```

Out[17]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x14f9624b0>



Discussion

The KNN model described in this notebook is able to predict the income of an individual based on the described categorical features with an accuracy of ~80% as seen in the training and test scores.

It was expected that selected categorical features would influence the income range for individuals, particularly those related to occupation and education level.

With above histograms, we notice that our KNN model predicts more individuals to have income that is less than 50K and predict less individuals to have more than 50K income, comparint to the actual results.

These findings support the notion that specific socioeconomic characteristics of individuals have a direct influence on the individual's income level.

However, this analysis opens the question on how each individual feature affects the model. Therefore, further deep-dive could better inform if all features have a significant influence on the model's ability to predict accurately. Additional numerical features, such as age and hours-per-week are likely to improve the model training process and could be evaluated as well.

References

- Becker, B. & Kohavi, R. (1996). Adult [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20.
- Kolhatkar, V. UBC Master of Data Science program, 2024-25, DSCI 571 Supervised Learning I.
- Ostblom, J. UBC Master of Data Science program, 2024-25, DSCI 573 Feature and Model Selection.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.