Set up modules/data

In [120…
```
import numpy as np
import pandas as pd
import math
import statsmodels.api as sm

pd.options.mode.copy_on_write = True
import random
from matplotlib.pyplot import subplots
import matplotlib.pyplot as plt
import statsmodels.api as sm
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize)
```

In [121…
```
from ISLP import confusion_table
from ISLP.models import contrast
from sklearn.discriminant_analysis import \
     (LinearDiscriminantAnalysis as LDA,
      QuadraticDiscriminantAnalysis as QDA)
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import RocCurveDisplay
roc_curve_est = RocCurveDisplay.from_estimator
roc_curve_pred = RocCurveDisplay.from_predictions
import statsmodels.api as sm
```

Load data set

In [122…
```
UCL_key = pd.read_csv('../Data/key_stats.csv')

UCL_key
```

Out[122…

	player_name	club	position	minutes_played	match_played	goals	assists	dist
0	Courtois	Real Madrid	Goalkeeper	1230	13	0	0	
1	Vinícius Júnior	Real Madrid	Forward	1199	13	4	6	
2	Benzema	Real Madrid	Forward	1106	12	15	1	
3	Modrić	Real Madrid	Midfielder	1077	13	0	4	
4	Éder Militão	Real Madrid	Defender	1076	12	0	0	
...	...	...	...	...	...	...	...	
742	Gil Dias	Benfica	Midfielder	1	1	0	0	
743	Rodrigo Ribeiro	Sporting CP	Forward	1	1	0	0	
744	Cojocari	Sheriff	Defender	1	1	0	0	
745	Maouassa	Club Brugge	Defender	1	1	0	0	
746	Zesiger	Young Boys	Defender	1	1	0	0	

747 rows × 8 columns

In [123…

```
import pandas as pd
from functools import reduce

# --- Load all the CSVs ---
key_stats    = pd.read_csv('../Data/key_stats.csv')      # base
disciplinary = pd.read_csv('../Data/disciplinary.csv')
attacking    = pd.read_csv('../Data/attacking.csv')
attempts     = pd.read_csv('../Data/attempts.csv')
defending    = pd.read_csv('../Data/defending.csv')
distribution = pd.read_csv('../Data/distributon.csv')
goals        = pd.read_csv('../Data/goals.csv')          # skip goalkeeping.csv

# --- Helper: add a prefix to ALL non-key columns so names are unique ---
def prefix_cols(df, prefix):
    return df.rename(columns={
        col: f'{prefix}_{col}' for col in df.columns if col != 'player_name'
    })

# Give each dataframe its own column namespace
key_stats_p    = prefix_cols(key_stats,    'ks')
disciplinary_p = prefix_cols(disciplinary, 'disc')
attacking_p    = prefix_cols(attacking,    'att')
```

```
attempts_p     = prefix_cols(attempts,     'attmpt')
defending_p    = prefix_cols(defending,    'def')
distribution_p = prefix_cols(distribution, 'dist')
goals_p        = prefix_cols(goals,        'goal')

# --- Put them into a list ---
datasets = [
    key_stats_p,
    disciplinary_p,
    attacking_p,
    attempts_p,
    defending_p,
    distribution_p,
    goals_p,
]

# --- Merge everything on player_name using outer join ---
mega_UCL = reduce(
    lambda left, right: pd.merge(left, right, on='player_name', how='outer'),
    datasets
)

# Optional: fill missing values
mega_UCL = mega_UCL.fillna(0)
mega_UCL
```

Out[123…

	player_name	ks_club	ks_position	ks_minutes_played	ks_match_played	ks_goals	k
0	Aaronson	Salzburg	Midfielder	715.0	8.0	0.0	
1	Abubakari	Malmö	Forward	116.0	4.0	0.0	
2	Acuña	Sevilla	Defender	379.0	5.0	0.0	
3	Adams	Leipzig	Midfielder	292.0	5.0	0.0	
4	Adamu	Salzburg	Forward	231.0	8.0	1.0	
…	…	…	…	…	…	…	
998	Óscar Rodríguez	Sevilla	Midfielder	22.0	1.0	0.0	
999	Čolak	Malmö	Forward	500.0	6.0	0.0	
1000	Šeško	Salzburg	Forward	234.0	6.0	0.0	
1001	Šimić	Salzburg	Forward	4.0	1.0	0.0	
1002	Škriniar	Inter	Defender	720.0	8.0	1.0	

1003 rows × 65 columns

Logistic Regression:

In [124…
```
megaUCL = mega_UCL.drop_duplicates(subset=['player_name', 'ks_club']) #gets rid of

megaUCL
```

Out[124…

	player_name	ks_club	ks_position	ks_minutes_played	ks_match_played	ks_goals	k
0	Aaronson	Salzburg	Midfielder	715.0	8.0	0.0	
1	Abubakari	Malmö	Forward	116.0	4.0	0.0	
2	Acuña	Sevilla	Defender	379.0	5.0	0.0	
3	Adams	Leipzig	Midfielder	292.0	5.0	0.0	
4	Adamu	Salzburg	Forward	231.0	8.0	1.0	
…	…	…	…	…	…	…	
998	Óscar Rodríguez	Sevilla	Midfielder	22.0	1.0	0.0	
999	Čolak	Malmö	Forward	500.0	6.0	0.0	
1000	Šeško	Salzburg	Forward	234.0	6.0	0.0	
1001	Šimić	Salzburg	Forward	4.0	1.0	0.0	
1002	Škriniar	Inter	Defender	720.0	8.0	1.0	

748 rows × 65 columns

In [125…
```
megaUCL[megaUCL['player_name'].duplicated()]['player_name'].unique() #shows players
```

Out[125…
```
array(['Camara', 'Correa', 'Danilo', 'Diallo', 'Fernando', 'Henderson',
       'Herrera', 'João Mário', 'Karavaev', 'Martínez', 'Mendy', 'Onana',
       'Peña', 'Sarr', 'Steffen'], dtype=object)
```

In [126…
```
Dupes = mega_UCL[mega_UCL['player_name'].duplicated()]['player_name'].unique()  #sh

#for name in Dupes:
    #print("\n=== Rows for:", name, "===")
    #display(mega_UCL_nodup[mega_UCL_nodup['player_name'] == name])
```

Find stastically significant features in data set

In [127…
```
print(UCL_key.columns)
print(UCL_key['position'].unique().tolist())
```

```
Index(['player_name', 'club', 'position', 'minutes_played', 'match_played',
       'goals', 'assists', 'distance_covered'],
      dtype='object')
['Goalkeeper', 'Forward', 'Midfielder', 'Defender']
```

In [128…
```
UCL_key = UCL_key.drop(UCL_key[UCL_key['position'] == 'Goalkeeper'].index)
UCL_key = UCL_key.drop(UCL_key[UCL_key['position'] == 'Midfielder'].index)
```

In [129…
```
UCL_key.loc[UCL_key['position']=="Defender"] = 0
UCL_key.loc[UCL_key['position']=="Forward"] = 1

X_UCL_key = UCL_key.drop(columns=['player_name', 'position', 'club'], axis=1)
Y_UCL_key = UCL_key['position']
X_UCL_key['intercept'] = np.ones(UCL_key.shape[0])
```

In [130…
```
glm = sm.GLM(Y_UCL_key,
             X_UCL_key,
             family=sm.families.Binomial())
results = glm.fit()
summarize(results)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[130], line 1
----> 1 glm = sm.GLM(Y_UCL_key,
      2              X_UCL_key,
      3              family=sm.families.Binomial())
      4 results = glm.fit()
      5 summarize(results)

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\genmod\general
ized_linear_model.py:326, in GLM.__init__(self, endog, exog, family, offset, exposur
e, freq_weights, var_weights, missing, **kwargs)
    323 self.freq_weights = freq_weights
    324 self.var_weights = var_weights
--> 326 super().__init__(endog, exog, missing=missing,
    327                            offset=offset, exposure=exposure,
    328                            freq_weights=freq_weights,
    329                            var_weights=var_weights, **kwargs)
    330 self._check_inputs(family, self.offset, self.exposure, self.endog,
    331                    self.freq_weights, self.var_weights)
    332 if offset is None:

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\model.py:
270, in LikelihoodModel.__init__(self, endog, exog, **kwargs)
    269 def __init__(self, endog, exog=None, **kwargs):
--> 270     super().__init__(endog, exog, **kwargs)
    271     self.initialize()

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\model.py:
95, in Model.__init__(self, endog, exog, **kwargs)
     93 missing = kwargs.pop('missing', 'none')
     94 hasconst = kwargs.pop('hasconst', None)
---> 95 self.data = self._handle_data(endog, exog, missing, hasconst,
     96                                **kwargs)
     97 self.k_constant = self.data.k_constant
     98 self.exog = self.data.exog

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\model.py:
135, in Model._handle_data(self, endog, exog, missing, hasconst, **kwargs)
    134 def _handle_data(self, endog, exog, missing, hasconst, **kwargs):
--> 135     data = handle_data(endog, exog, missing, hasconst, **kwargs)
    136     # kwargs arrays could have changed, easier to just attach here
    137     for key in kwargs:

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\data.py:6
75, in handle_data(endog, exog, missing, hasconst, **kwargs)
    672     exog = np.asarray(exog)
    674 klass = handle_data_class_factory(endog, exog)
--> 675 return klass(endog, exog=exog, missing=missing, hasconst=hasconst,
    676              **kwargs)

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\data.py:8
4, in ModelData.__init__(self, endog, exog, missing, hasconst, **kwargs)
     82     self.orig_endog = endog
     83     self.orig_exog = exog
---> 84     self.endog, self.exog = self._convert_endog_exog(endog, exog)
```

```
     86 self.const_idx = None
     87 self.k_constant = 0

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\data.py:5
09, in PandasData._convert_endog_exog(self, endog, exog)
    507 exog = exog if exog is None else np.asarray(exog)
    508 if endog.dtype == object or exog is not None and exog.dtype == object:
--> 509     raise ValueError("Pandas data cast to numpy dtype of object. "
    510                      "Check input data with np.asarray(data).")
    511 return super()._convert_endog_exog(endog, exog)

ValueError: Pandas data cast to numpy dtype of object. Check input data with np.asar
ray(data).
```

Run model to see what gives test data

In [ ]:

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[99], line 1
----> 1 model = sm.MNLogit(y, X)
      2 results = model.fit(method='newton', maxiter=200, disp=False)

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\discrete\discr
ete_model.py:2990, in MNLogit.__init__(self, endog, exog, check_rank, **kwargs)
   2989 def __init__(self, endog, exog, check_rank=True, **kwargs):
-> 2990     super().__init__(endog, exog, check_rank=check_rank, **kwargs)
   2992     # Override cov_names since multivariate model
   2993     yname = self.endog_names

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\discrete\discr
ete_model.py:475, in BinaryModel.__init__(self, endog, exog, offset, check_rank, **k
wargs)
    472 def __init__(self, endog, exog, offset=None, check_rank=True, **kwargs):
    473     # unconditional check, requires no extra kwargs added by subclasses
    474     self._check_kwargs(kwargs)
--> 475     super().__init__(endog, exog, offset=offset, check_rank=check_rank,
    476                      **kwargs)
    477     if not issubclass(self.__class__, MultinomialModel):
    478         if not np.all((self.endog >= 0) & (self.endog <= 1)):

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\discrete\discr
ete_model.py:185, in DiscreteModel.__init__(self, endog, exog, check_rank, **kwargs)
    183 def __init__(self, endog, exog, check_rank=True, **kwargs):
    184     self._check_rank = check_rank
--> 185     super().__init__(endog, exog, **kwargs)
    186     self.raise_on_perfect_prediction = False  # keep for backwards compat
    187     self.k_extra = 0

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\model.py:
270, in LikelihoodModel.__init__(self, endog, exog, **kwargs)
    269 def __init__(self, endog, exog=None, **kwargs):
--> 270     super().__init__(endog, exog, **kwargs)
    271     self.initialize()

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\model.py:
95, in Model.__init__(self, endog, exog, **kwargs)
     93 missing = kwargs.pop('missing', 'none')
     94 hasconst = kwargs.pop('hasconst', None)
---> 95 self.data = self._handle_data(endog, exog, missing, hasconst,
     96                                **kwargs)
     97 self.k_constant = self.data.k_constant
     98 self.exog = self.data.exog

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\discrete\discr
ete_model.py:707, in MultinomialModel._handle_data(self, endog, exog, missing, hasco
nst, **kwargs)
    704     ynames = dict(zip(range(endog_dummies.shape[1]), ynames))
    706 self._ynames_map = ynames
--> 707 data = handle_data(endog_dummies, exog, missing, hasconst, **kwargs)
    708 data.ynames = yname  # overwrite this to single endog name
    709 data.orig_endog = endog
```

```
File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\data.py:6
75, in handle_data(endog, exog, missing, hasconst, **kwargs)
    672     exog = np.asarray(exog)
    674 klass = handle_data_class_factory(endog, exog)
--> 675 return klass(endog, exog=exog, missing=missing, hasconst=hasconst,
    676              **kwargs)

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\data.py:8
4, in ModelData.__init__(self, endog, exog, missing, hasconst, **kwargs)
     82     self.orig_endog = endog
     83     self.orig_exog = exog
---> 84     self.endog, self.exog = self._convert_endog_exog(endog, exog)
     86 self.const_idx = None
     87 self.k_constant = 0

File c:\Users\iacom\anaconda3\envs\MAIN\Lib\site-packages\statsmodels\base\data.py:5
09, in PandasData._convert_endog_exog(self, endog, exog)
    507 exog = exog if exog is None else np.asarray(exog)
    508 if endog.dtype == object or exog is not None and exog.dtype == object:
--> 509     raise ValueError("Pandas data cast to numpy dtype of object. "
    510                      "Check input data with np.asarray(data).")
    511 return super()._convert_endog_exog(endog, exog)

ValueError: Pandas data cast to numpy dtype of object. Check input data with np.asar
ray(data).
```

Run k-fold to test data since less than 1000 observations

In [ ]:

K-means Clustering

Normalize data

In [ ]:

Run k-means, we can discuss what best one is or switch to hierarchial

In [ ]:
```
#I was thinking 4 for 4 positions or we can research how many football archetypes t
```

Random Forest

Run the random forest

In [ ]:

Compare to logistic regression

In [ ]: