CENTENNIAL
COLLEGE

BA706 - Applied Analytic Modeling

Professional Revision

Students:

- Aishwarya Dhairya Shah
- Juan Pablo Tamayo Villa
- Max Sergio Causso Fretel
- Urielle Silvana Megnokam Fotso

Table of Contents

1. Executive Summary

This project aims to assess and enhance the company's hiring decisions through data-driven insights derived from predictive modeling. By analyzing key candidate attributes—including experience, education, skill and personality scores, and recruitment strategies—we developed and evaluated multiple machine learning models to identify the factors that most significantly influence successful hiring outcomes.

Among the models tested—Decision Trees, Logistic Regression, and Neural Networks—the Forward Logistic Regression model emerged as the most interpretable and reliable, achieving an accuracy of approximately **90.5%**. Key predictors of hiring success include higher education levels (Master's or PhD), strong performance on skill and personality assessments, and recruitment through

moderate or conservative strategies, years of experience. The findings also highlight a potential bias against early-career candidates, which may merit further review for compliance with equity and inclusion standards.

2. Business Objective

This project involves a comprehensive analysis of the characteristics and qualifications of recent hires to assess the effectiveness of the company's recruitment strategy in attracting talent aligned with its organizational goals and culture.

Evaluating hiring practices is critical, as selecting the wrong candidates can lead to significant consequences—such as higher turnover costs, wasted investments in training and development, disruptions to team cohesion, and a decline in overall productivity and efficiency.

Furthermore, this analysis can help determine whether the company's hiring practices inadvertently involve discriminatory bias. According to the *Ontario Human Rights Code, R.S.O. 1990, c. H.19*, individuals or organizations that infringe upon the right to equal treatment in employment may face penalties of up to $25,000.

By understanding whether current hiring decisions are delivering the intended employee profile, the company can make evidence-based improvements to its talent acquisition process, enhance workforce performance, and strengthen alignment with its long-term strategic objectives.

3. Data Overview

Below is the detail of original dataset.

Variable	Description	Data type
Age	Age of the candidate	Interval
Gender	Gender of the candidate - Male (0) or Female (1)	Binary
EducationLevel	Highest level of education attained by the candidate: 1: Bachelor's (Type 1) 2: Bachelor's (Type 2) 3: Master's 4: PhD	Nominal

ExperienceYears	Number of years of professional experience. 0 to 15 years.	Interval
PreviousCompanies	Number of previous companies where the candidate has worked. 1 to 5 companies.	Interval
DistanceFromCompany	Distance in kilometers from the candidate's residence to the hiring company. 1 to 50 kilometers.	Interval
InterviewScore	Score achieved by the candidate in the interview process. 0 to 100.	Interval
SkillScore	Assessment score of the candidate's technical skills. 0 to 100.	Interval
PersonalityScore	Evaluation score of the candidate's personality traits. 0 to 100.	Interval
RecruitmentStrategy	Strategy adopted by the hiring team for recruitment. • 1: Aggressive • 2: Moderate • 3: Conservative	Nominal
HiringDecision	Outcome of the hiring decision. • 0: Not hired • 1: Hired	Binary

The above dataset has 1500 columns and 11 rows.

We have recoded the variables for Education level wherein we have merged Bachelor’s type 1 and Bachelor's 2 as L2. We created categories for age as G[20-27], G[28-40] and G[41-50]. Further, we created categories and grouped years of experience into Y[0], Y[1-5], Y[6-10], Y[11-15].

New dataset is as below:

Variable	**Description**	**Data type**
Age	Age of the candidate	Interval
GroupAge	Variable that groups the ages of the candidates: • G[20-27]: candidates between the ages of 20 and 27. • G[28-40]: candidates between the ages of 28 and 40. • G[41-50]: candidates between the ages of 41 and 50.	Nominal

Gender	Gender of the candidate - Male (0) or Female (1)	Binary
EducationLevel	Highest level of education attained by the candidate: L1: Bachelor's (Type 1) L2: Bachelor's (Type 2) L3: Master's L4: PhD	Nominal
ExperienceYears	Number of years of professional experience. 0 to 15 years.	Interval
FirstJob	Indicates whether the candidate is applying for his or her first job (1) or not (0).	Binary
GroupYOE	Group the years of experience of each candidate: • Y[0]: People with 0 years of experience. • Y[1-5]: People between 1 and 5 years of experience. • Y[6-10]: People between 6 and 10 years of experience. • Y[11-15]: People between 11 and 15 years of experience.	Nominal
PreviousCompanies	Number of previous companies where the candidate has worked. 1 to 5 companies.	Interval
DistanceFromCompany	Distance in kilometers from the candidate's residence to the hiring company. 1 to 50 kilometers.	Interval
InterviewScore	Score achieved by the candidate in the interview process. 0 to 100.	Interval
SkillScore	Assessment score of the candidate's technical skills. 0 to 100.	Interval
PersonalityScore	Evaluation score of the candidate's personality traits. 0 to 100.	Interval
RecruitmentStrategy	Strategy adopted by the hiring team for recruitment. • S1: Aggressive • S2: Moderate • S3: Conservative	Nominal
HiringDecision	Outcome of the hiring decision. • 0: Not hired	Binary

	• 1: Hired	

The updated dataset has 1500 rows and 14 columns

4. Methodology

In this analysis, we explored three types of models to predict hiring decisions: Decision Trees, Logistic Regression (Full, Forward, Backward, Stepwise), and Neural Network (Multilayer Perceptron). Each model was fine-tuned to optimize performance, and all datasets were partitioned using a 50/50 train-test split to ensure fair evaluation. Feature engineering included encoding categorical variables using `get_dummies()` for inputs such as GroupAge, GroupYOE (Years of Experience), REP_EducationLevel, and RecruitmentStrategy. This transformation was necessary to ensure compatibility with algorithms that require numerical inputs.

4.1 Decision Tree Models

We evaluated three tree-based variants: Maximum Tree, ASE Tree (based on Average Squared Error), and Misclassification Tree. The best performing model among these was the ASE Tree, which achieved an ASE of 0.13021 with an optimal depth of 5 and 27 leaves. From the Gini index analysis, we identified two high-performing candidate profiles. Profile 1 includes candidates with 1–5 years of group experience, interview scores above 82.5, personality scores above 62.5, and who were recruited via RecruitmentStrategy_3. Profile 2 includes candidates with personality scores above 60.5, skill scores above 18.5, and recruitment via RecruitmentStrategy_3. See visualization of the best performing tree below

4.2 Logistic Regression Models

We ran four logistic regression models: Full, Forward, Backward, and Stepwise. The Forward Selection model had the lowest ASE (0.089437) and was thus identified as the best-performing regression model. The most influential features in this model included Age, Gender, Experience Years, First Job, Previous Companies, Distance From Company, Interview Score, Skill Score, Personality Score, Group YOE (1–5 years), REP_EducationLevel_L3 and L4, and RecruitmentStrategy_2 and _3. Below are the odds ratio of selected features

Odds Ratio Interpretation (Top Features):

Feature	Interpretation
Age	0.8% decrease in hiring likelihood per additional year
Gender (Female)	3.5times higher odds of being hired (vs. Male)
ExperienceYears	9% decrease in odds per additional year

FirstJob (Yes)	97.8% decrease in hiring likelihood
PreviousCompanies	17.8% increase in odds per additional company
DistanceFromCompany	1.9% decrease in odds per unit distance
SkillScore	4.5% increase in hiring likelihood per point
InterviewScore	3.5% increase in hiring likelihood per point
PersonalityScore	3.1% increase in hiring likelihood per point
GroupYOE (1–5)	95.2% decrease in odds
REP_EducationLevel_L3	13 times higher odds of being hired
REP_EducationLevel_L4	18.8 times higher odds of being hired
RecruitmentStrategy_2	99.5% higher odds of being hired
RecruitmentStrategy_3	99.6% higher odds of being hired

4.3 Neural Network

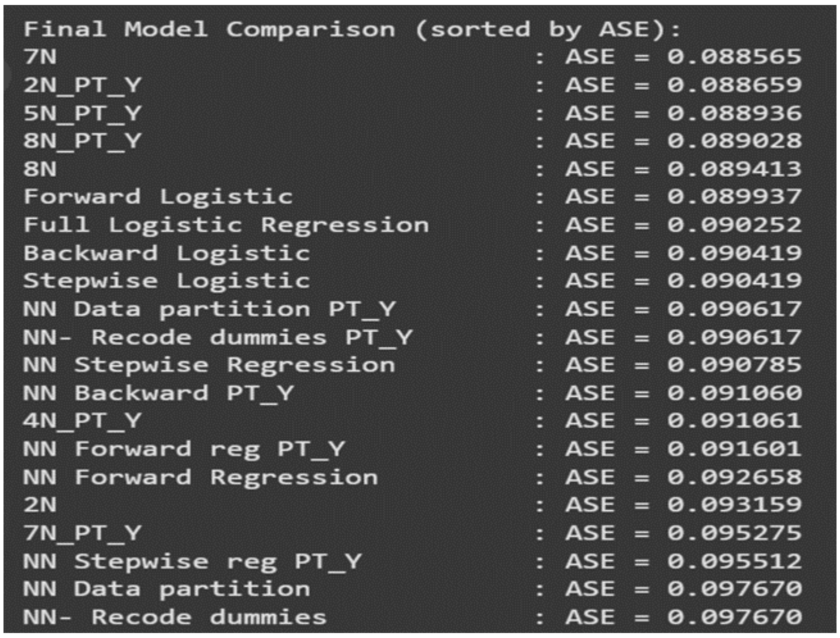
To capture the non-linear relationships in the data, we implemented a neural network using a Multilayer Perceptron (MLP) architecture. The model was trained using various forms of the dataset to evaluate how changes in data quality and structure impacted performance.

Since neural networks lack an inherent mechanism for input feature selection, we complemented our design with insights from logistic regression models to guide variable selection. We used ReLU and Sigmoid as activation functions, Binary Cross-Entropy as the loss function, and Adam as the optimizer.

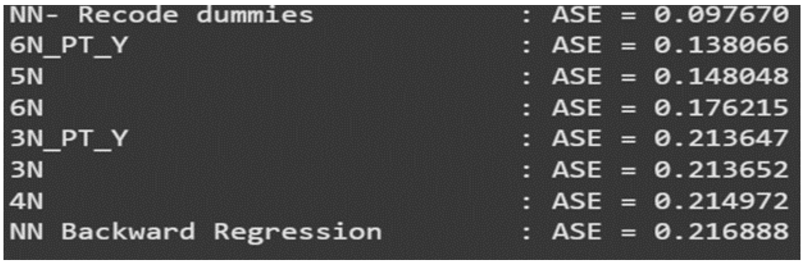
To fine-tune the model, we experimented with different numbers of hidden units, ranging from 2 to 8, and set the number of iterations to 100 epochs. The dataset used was transformed using dummy variables and appropriately partitioned to ensure consistency. All models were evaluated with pre-training enabled, allowing the algorithm to establish preliminary learning patterns before full optimization.

After testing all configurations, the best-performing neural network featured 7 hidden units and achieved the lowest ASE of 0.088565. While this model demonstrated the strongest predictive performance, it was not selected as the final model due to its limited interpretability.

Below is the model comparison for all the neural networks ran:



```
Final Model Comparison (sorted by ASE):
7N                                : ASE = 0.088565
2N_PT_Y                           : ASE = 0.088659
5N_PT_Y                           : ASE = 0.088936
8N_PT_Y                           : ASE = 0.089028
8N                                : ASE = 0.089413
Forward Logistic                  : ASE = 0.089937
Full Logistic Regression          : ASE = 0.090252
Backward Logistic                 : ASE = 0.090419
Stepwise Logistic                 : ASE = 0.090419
NN Data partition PT_Y            : ASE = 0.090617
NN- Recode dummies PT_Y           : ASE = 0.090617
NN Stepwise Regression            : ASE = 0.090785
NN Backward PT_Y                  : ASE = 0.091060
4N_PT_Y                           : ASE = 0.091061
NN Forward reg PT_Y               : ASE = 0.091601
NN Forward Regression             : ASE = 0.092658
2N                                : ASE = 0.093159
7N_PT_Y                           : ASE = 0.095275
NN Stepwise reg PT_Y              : ASE = 0.095512
NN Data partition                 : ASE = 0.097670
NN- Recode dummies                : ASE = 0.097670
```



```
NN- Recode dummies                : ASE = 0.097670
6N_PT_Y                           : ASE = 0.138066
5N                                : ASE = 0.148048
6N                                : ASE = 0.176215
3N_PT_Y                           : ASE = 0.213647
3N                                : ASE = 0.213652
4N                                : ASE = 0.214972
NN Backward Regression            : ASE = 0.216888
```

5. Model Performance Comparison

Below is a summary of model performance based on ASE:

Model	ASE	Comments
Neural Network (7 Units)	0.08857	Best ASE but low interpretability
Forward Logistic Regression	0.089437	Best interpretable model
ASE Decision Tree	0.13021	High performance, some interpretability

In addition to ASE, ROC AUC was used to confirm our selection. The 4N had the highest ROC followed by the 4 Logistic Regressions with the same value (0.9054). So, the Forward Regression model was selected as the final model. It offers the best balance between performance and transparency, making it ideal for real-world deployment.

6. Key Insights

The results of the models allow us to identify what are the key features that drive hiring decisions inside the company, the most important are:

Years of experience: The company tends to favor candidates with **more extensive and diverse experience**. Applicants with no prior job experience or only 1 to 5 years of experience see a significant drop in their likelihood of being hired. However, candidates who have worked for multiple companies are more likely to be selected, suggesting that the company values both breadth and depth of experience, rather than just the number of years alone

Test Scores: When the scores of the skill, interview and personality tests increase the likelihood of getting hired slightly increase, this means that the company appreciates the results of these tests when deciding but doesn't decide solely based on the tests. The test that seems to be more important is the skill test.

Education Level: The chances of hiring increase when the applicant holds a master's or PhD, which shows how the company prefers candidates with higher education levels.

Recruitment Strategy: Candidates approached with a moderate and conservative recruitment strategy are much more likely to get hired that the ones approached with an aggressive strategy, this indicates that the aggressive strategy may be biased to not hire the applicants.

Overall, the results show how the company is hiring applicants with more experience measured in both years of experience and previous companies, the company also values knowledge and skills as they are more likely to hire applicants with higher education levels and good results on the skills tests.

7. Recommendations and next steps

The models provide us with insights on the characteristics that are driving the hiring decisions, however, these models alone cannot conclude if the hiring process is being successful or not, to correctly evaluate this we need to first understand what the hiring objectives of the company are and the type of candidates the company wants to attract.

In this case, the company is hiring candidates with more experience, so, if the positions the company needs to fill require experienced employees, we could say that the hiring process seems to be appropriate for the company.

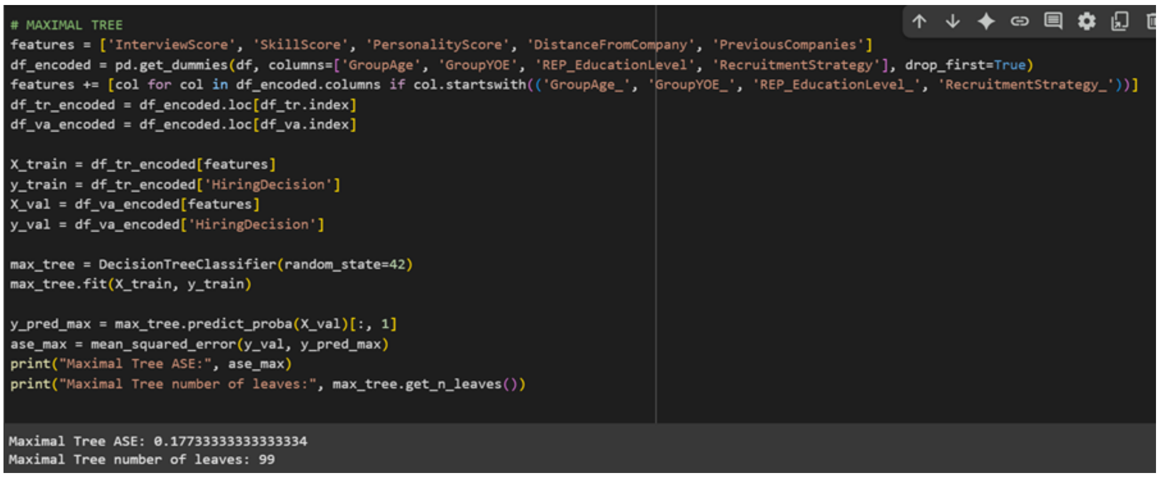
The next step that should be taken to improve the hiring decision analysis is to deeply understand the hiring needs and objectives inside the company and use the model's results to check if the hiring decisions are aligned with the objectives.

CENTENNIAL COLLEGE

APPENDIX

CODE SNIPPET

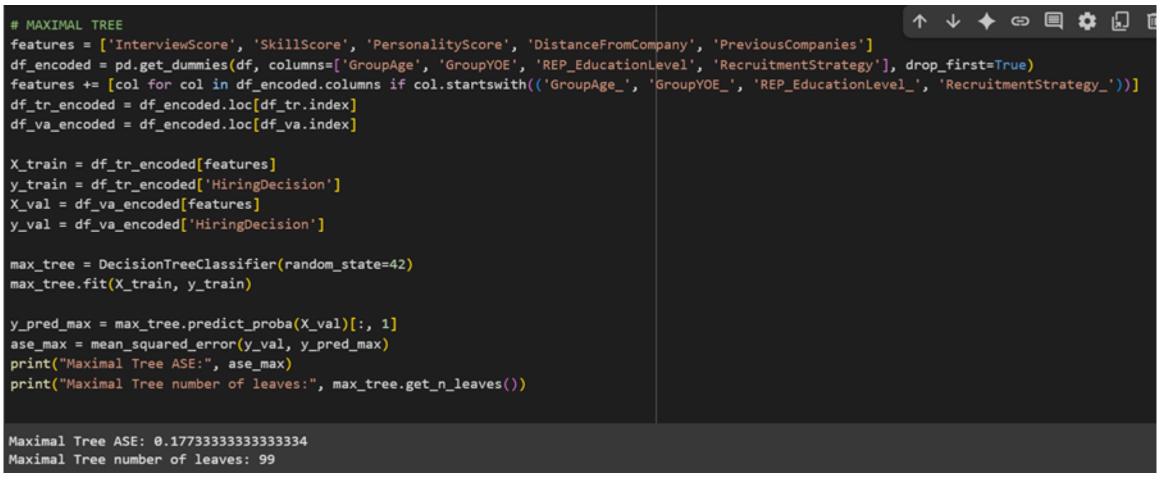
Maximum tree



```
# MAXIMAL TREE
features = ['InterviewScore', 'SkillScore', 'PersonalityScore', 'DistanceFromCompany', 'PreviousCompanies']
df_encoded = pd.get_dummies(df, columns=['GroupAge', 'GroupYOE', 'REP_EducationLevel', 'RecruitmentStrategy'], drop_first=True)
features += [col for col in df_encoded.columns if col.startswith(('GroupAge_', 'GroupYOE_', 'REP_EducationLevel_', 'RecruitmentStrategy_'))]
df_tr_encoded = df_encoded.loc[df_tr.index]
df_va_encoded = df_encoded.loc[df_va.index]

X_train = df_tr_encoded[features]
y_train = df_tr_encoded['HiringDecision']
X_val = df_va_encoded[features]
y_val = df_va_encoded['HiringDecision']

max_tree = DecisionTreeClassifier(random_state=42)
max_tree.fit(X_train, y_train)

y_pred_max = max_tree.predict_proba(X_val)[:, 1]
ase_max = mean_squared_error(y_val, y_pred_max)
print("Maximal Tree ASE:", ase_max)
print("Maximal Tree number of leaves:", max_tree.get_n_leaves())
```

```
Maximal Tree ASE: 0.17733333333333334
Maximal Tree number of leaves: 99
```

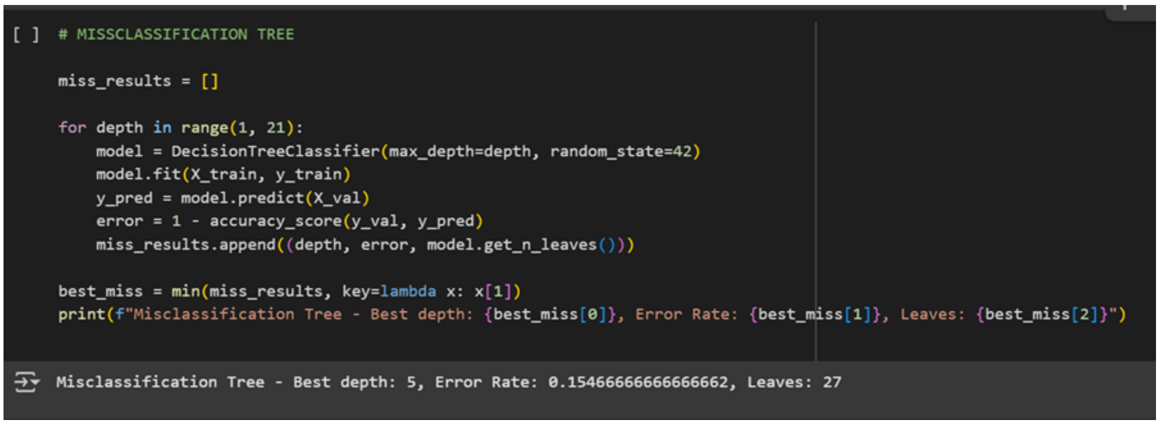
Ase tree



```
# MAXIMAL TREE
features = ['InterviewScore', 'SkillScore', 'PersonalityScore', 'DistanceFromCompany', 'PreviousCompanies']
df_encoded = pd.get_dummies(df, columns=['GroupAge', 'GroupYOE', 'REP_EducationLevel', 'RecruitmentStrategy'], drop_first=True)
features += [col for col in df_encoded.columns if col.startswith(('GroupAge_', 'GroupYOE_', 'REP_EducationLevel_', 'RecruitmentStrategy_'))]
df_tr_encoded = df_encoded.loc[df_tr.index]
df_va_encoded = df_encoded.loc[df_va.index]

X_train = df_tr_encoded[features]
y_train = df_tr_encoded['HiringDecision']
X_val = df_va_encoded[features]
y_val = df_va_encoded['HiringDecision']

max_tree = DecisionTreeClassifier(random_state=42)
max_tree.fit(X_train, y_train)

y_pred_max = max_tree.predict_proba(X_val)[:, 1]
ase_max = mean_squared_error(y_val, y_pred_max)
print("Maximal Tree ASE:", ase_max)
print("Maximal Tree number of leaves:", max_tree.get_n_leaves())
```

```
Maximal Tree ASE: 0.17733333333333334
Maximal Tree number of leaves: 99
```

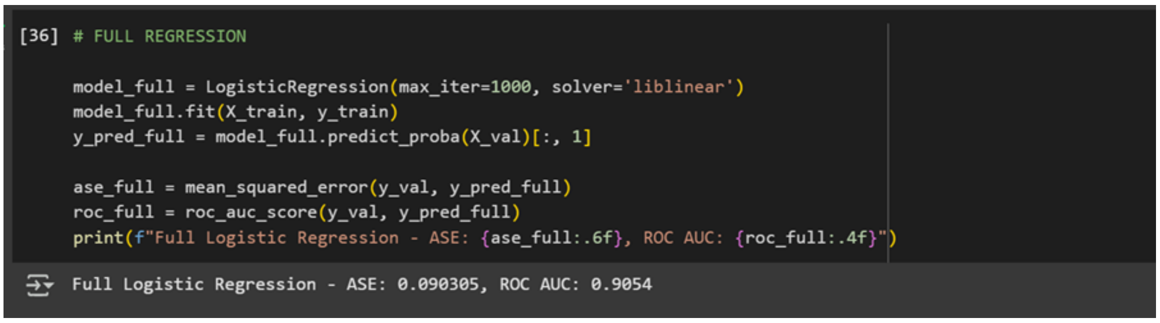
Misclassification tree



```
[ ]  # MISSCLASSIFICATION TREE

     miss_results = []

     for depth in range(1, 21):
         model = DecisionTreeClassifier(max_depth=depth, random_state=42)
         model.fit(X_train, y_train)
         y_pred = model.predict(X_val)
         error = 1 - accuracy_score(y_val, y_pred)
         miss_results.append((depth, error, model.get_n_leaves()))

     best_miss = min(miss_results, key=lambda x: x[1])
     print(f"Misclassification Tree - Best depth: {best_miss[0]}, Error Rate: {best_miss[1]}, Leaves: {best_miss[2]}")
```

```
Misclassification Tree - Best depth: 5, Error Rate: 0.15466666666666662, Leaves: 27
```

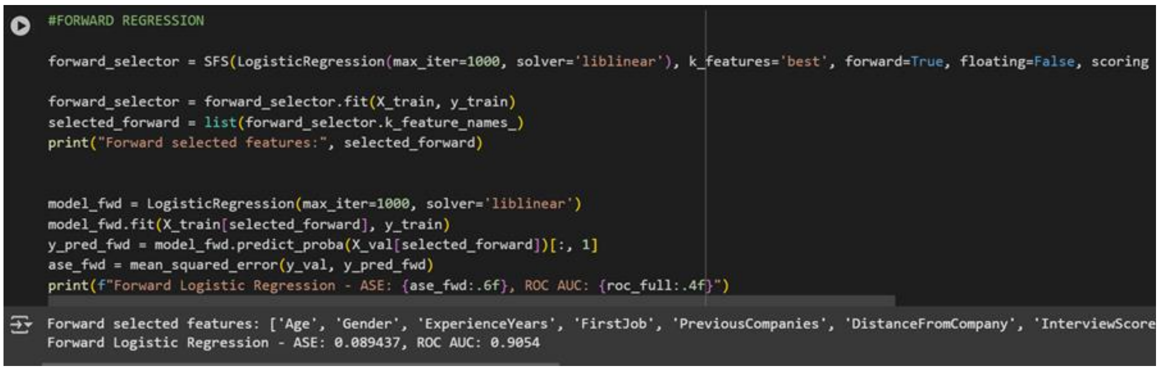
Full regression



```
[36] # FULL REGRESSION

     model_full = LogisticRegression(max_iter=1000, solver='liblinear')
     model_full.fit(X_train, y_train)
     y_pred_full = model_full.predict_proba(X_val)[:, 1]

     ase_full = mean_squared_error(y_val, y_pred_full)
     roc_full = roc_auc_score(y_val, y_pred_full)
     print(f"Full Logistic Regression - ASE: {ase_full:.6f}, ROC AUC: {roc_full:.4f}")
```

```
Full Logistic Regression - ASE: 0.090305, ROC AUC: 0.9054
```

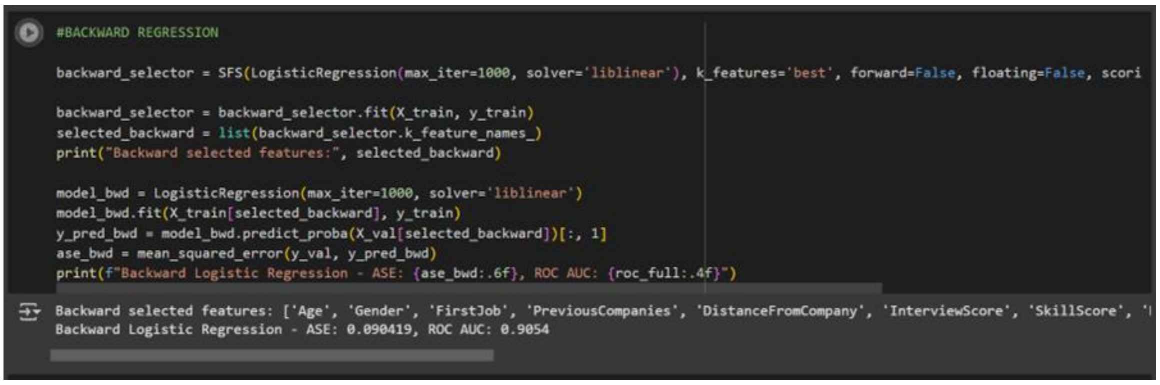
Forward regression



```
#FORWARD REGRESSION

forward_selector = SFS(LogisticRegression(max_iter=1000, solver='liblinear'), k_features='best', forward=True, floating=False, scoring

forward_selector = forward_selector.fit(X_train, y_train)
selected_forward = list(forward_selector.k_feature_names_)
print("Forward selected features:", selected_forward)


model_fwd = LogisticRegression(max_iter=1000, solver='liblinear')
model_fwd.fit(X_train[selected_forward], y_train)
y_pred_fwd = model_fwd.predict_proba(X_val[selected_forward])[:, 1]
ase_fwd = mean_squared_error(y_val, y_pred_fwd)
print(f"Forward Logistic Regression - ASE: {ase_fwd:.6f}, ROC AUC: {roc_full:.4f}")
```

```
Forward selected features: ['Age', 'Gender', 'ExperienceYears', 'FirstJob', 'PreviousCompanies', 'DistanceFromCompany', 'InterviewScore
Forward Logistic Regression - ASE: 0.089437, ROC AUC: 0.9054
```

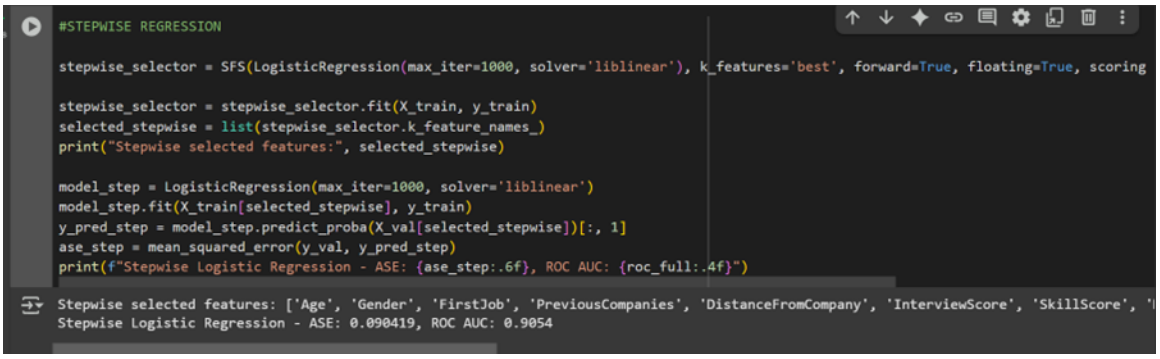
Backward regression



```
#BACKWARD REGRESSION

backward_selector = SFS(LogisticRegression(max_iter=1000, solver='liblinear'), k_features='best', forward=False, floating=False, scori

backward_selector = backward_selector.fit(X_train, y_train)
selected_backward = list(backward_selector.k_feature_names_)
print("Backward selected features:", selected_backward)

model_bwd = LogisticRegression(max_iter=1000, solver='liblinear')
model_bwd.fit(X_train[selected_backward], y_train)
y_pred_bwd = model_bwd.predict_proba(X_val[selected_backward])[:, 1]
ase_bwd = mean_squared_error(y_val, y_pred_bwd)
print(f"Backward Logistic Regression - ASE: {ase_bwd:.6f}, ROC AUC: {roc_full:.4f}")
```

```
Backward selected features: ['Age', 'Gender', 'FirstJob', 'PreviousCompanies', 'DistanceFromCompany', 'InterviewScore', 'SkillScore', '
Backward Logistic Regression - ASE: 0.090419, ROC AUC: 0.9054
```

Step wise regression

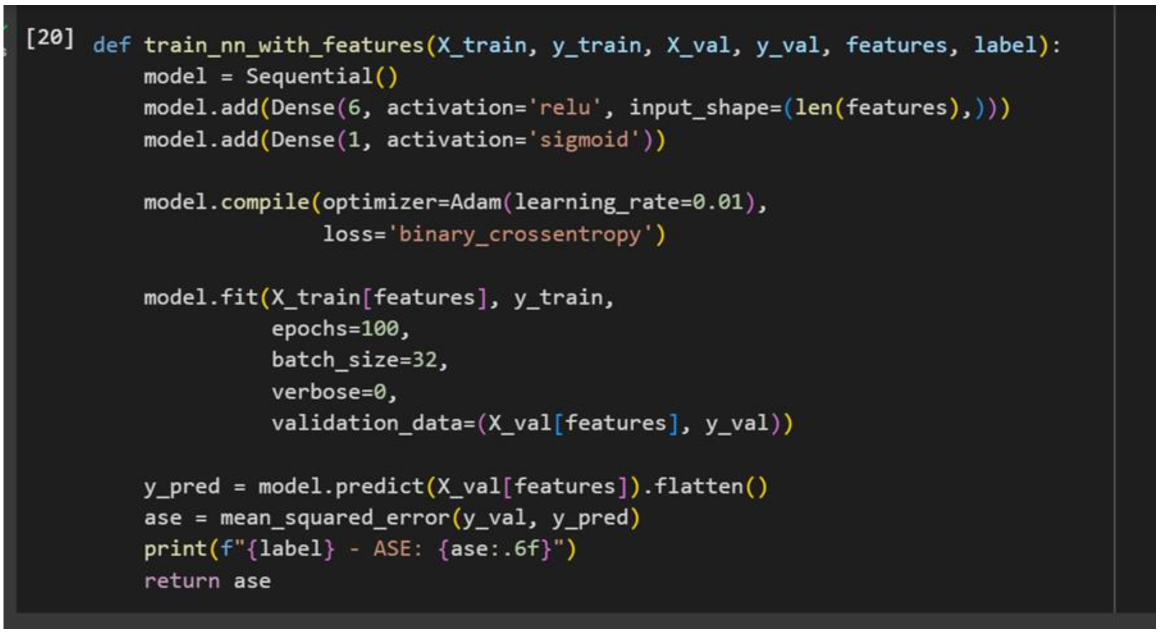


```
#STEPWISE REGRESSION

stepwise_selector = SFS(LogisticRegression(max_iter=1000, solver='liblinear'), k_features='best', forward=True, floating=True, scoring

stepwise_selector = stepwise_selector.fit(X_train, y_train)
selected_stepwise = list(stepwise_selector.k_feature_names_)
print("Stepwise selected features:", selected_stepwise)

model_step = LogisticRegression(max_iter=1000, solver='liblinear')
model_step.fit(X_train[selected_stepwise], y_train)
y_pred_step = model_step.predict_proba(X_val[selected_stepwise])[:, 1]
ase_step = mean_squared_error(y_val, y_pred_step)
print(f"Stepwise Logistic Regression - ASE: {ase_step:.6f}, ROC AUC: {roc_full:.4f}")
```

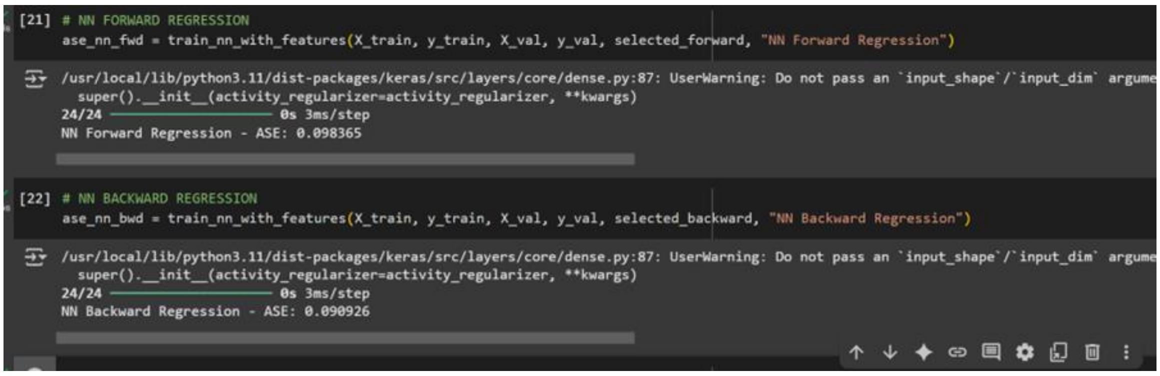
```
Stepwise selected features: ['Age', 'Gender', 'FirstJob', 'PreviousCompanies', 'DistanceFromCompany', 'InterviewScore', 'SkillScore', '
Stepwise Logistic Regression - ASE: 0.090419, ROC AUC: 0.9054
```

CENTENNIAL
COLLEGE

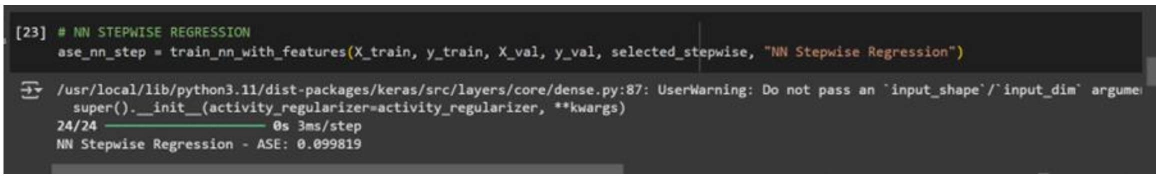
NN Forward regression, backward and stepwise regression



```
[20] def train_nn_with_features(X_train, y_train, X_val, y_val, features, label):
         model = Sequential()
         model.add(Dense(6, activation='relu', input_shape=(len(features),)))
         model.add(Dense(1, activation='sigmoid'))

         model.compile(optimizer=Adam(learning_rate=0.01),
                       loss='binary_crossentropy')

         model.fit(X_train[features], y_train,
                   epochs=100,
                   batch_size=32,
                   verbose=0,
                   validation_data=(X_val[features], y_val))

         y_pred = model.predict(X_val[features]).flatten()
         ase = mean_squared_error(y_val, y_pred)
         print(f"{label} - ASE: {ase:.6f}")
         return ase
```

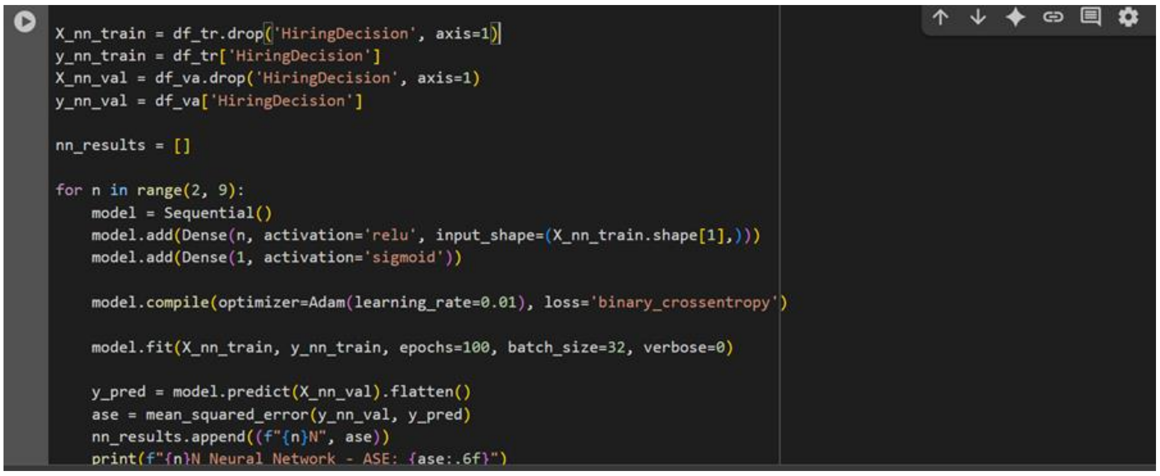


```
[21] # NN FORWARD REGRESSION
     ase_nn_fwd = train_nn_with_features(X_train, y_train, X_val, y_val, selected_forward, "NN Forward Regression")

     /usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argume
       super().__init__(activity_regularizer=activity_regularizer, **kwargs)
     24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
     NN Forward Regression - ASE: 0.098365

[22] # NN BACKWARD REGRESSION
     ase_nn_bwd = train_nn_with_features(X_train, y_train, X_val, y_val, selected_backward, "NN Backward Regression")

     /usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argume
       super().__init__(activity_regularizer=activity_regularizer, **kwargs)
     24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
     NN Backward Regression - ASE: 0.090926
```

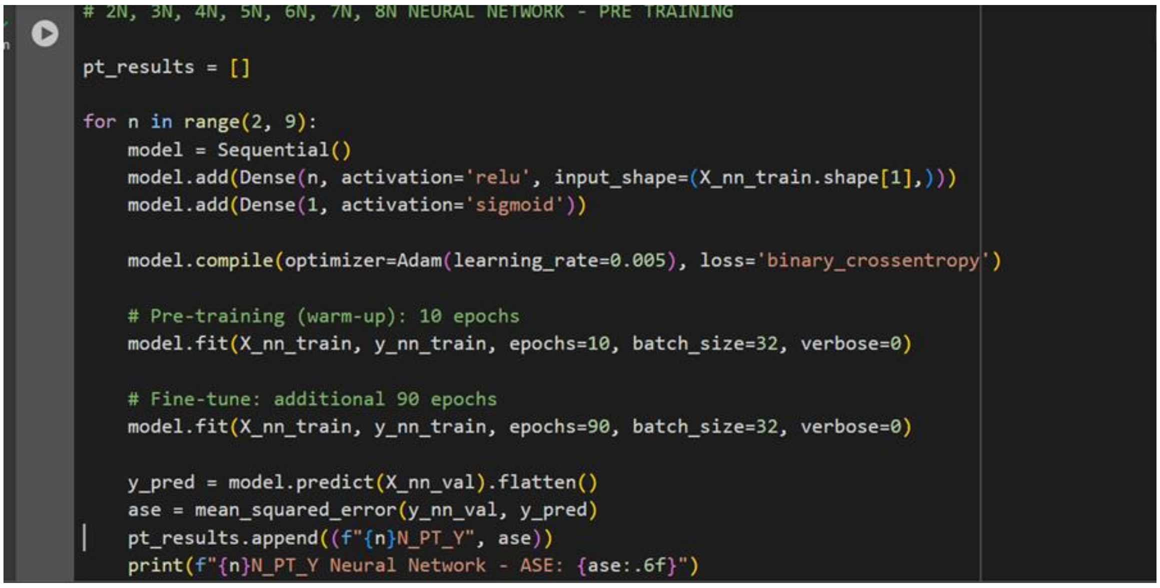


```
[23] # NN STEPWISE REGRESSION
     ase_nn_step = train_nn_with_features(X_train, y_train, X_val, y_val, selected_stepwise, "NN Stepwise Regression")

     /usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argume
       super().__init__(activity_regularizer=activity_regularizer, **kwargs)
     24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
     NN Stepwise Regression - ASE: 0.099819
```

NN with 2 to 8 hidden units and with pre training

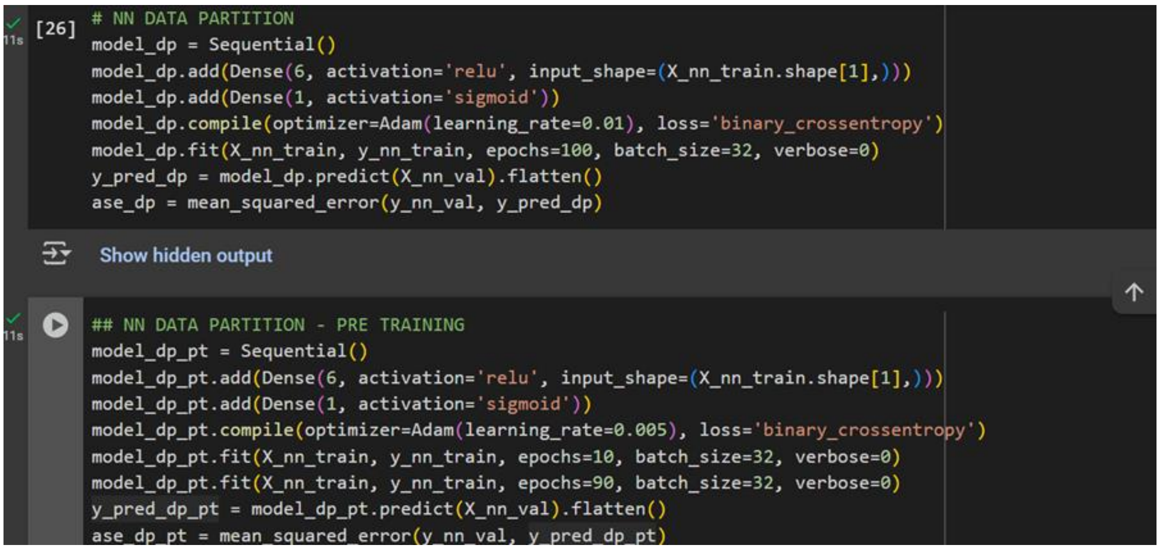


```python
X_nn_train = df_tr.drop('HiringDecision', axis=1)
y_nn_train = df_tr['HiringDecision']
X_nn_val = df_va.drop('HiringDecision', axis=1)
y_nn_val = df_va['HiringDecision']

nn_results = []

for n in range(2, 9):
    model = Sequential()
    model.add(Dense(n, activation='relu', input_shape=(X_nn_train.shape[1],)))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer=Adam(learning_rate=0.01), loss='binary_crossentropy')

    model.fit(X_nn_train, y_nn_train, epochs=100, batch_size=32, verbose=0)

    y_pred = model.predict(X_nn_val).flatten()
    ase = mean_squared_error(y_nn_val, y_pred)
    nn_results.append((f"{n}N", ase))
    print(f"{n}N Neural Network - ASE: {ase:.6f}")
```



```python
# 2N, 3N, 4N, 5N, 6N, 7N, 8N NEURAL NETWORK - PRE TRAINING

pt_results = []

for n in range(2, 9):
    model = Sequential()
    model.add(Dense(n, activation='relu', input_shape=(X_nn_train.shape[1],)))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer=Adam(learning_rate=0.005), loss='binary_crossentropy')

    # Pre-training (warm-up): 10 epochs
    model.fit(X_nn_train, y_nn_train, epochs=10, batch_size=32, verbose=0)

    # Fine-tune: additional 90 epochs
    model.fit(X_nn_train, y_nn_train, epochs=90, batch_size=32, verbose=0)

    y_pred = model.predict(X_nn_val).flatten()
    ase = mean_squared_error(y_nn_val, y_pred)
    pt_results.append((f"{n}N_PT_Y", ase))
    print(f"{n}N_PT_Y Neural Network - ASE: {ase:.6f}")
```

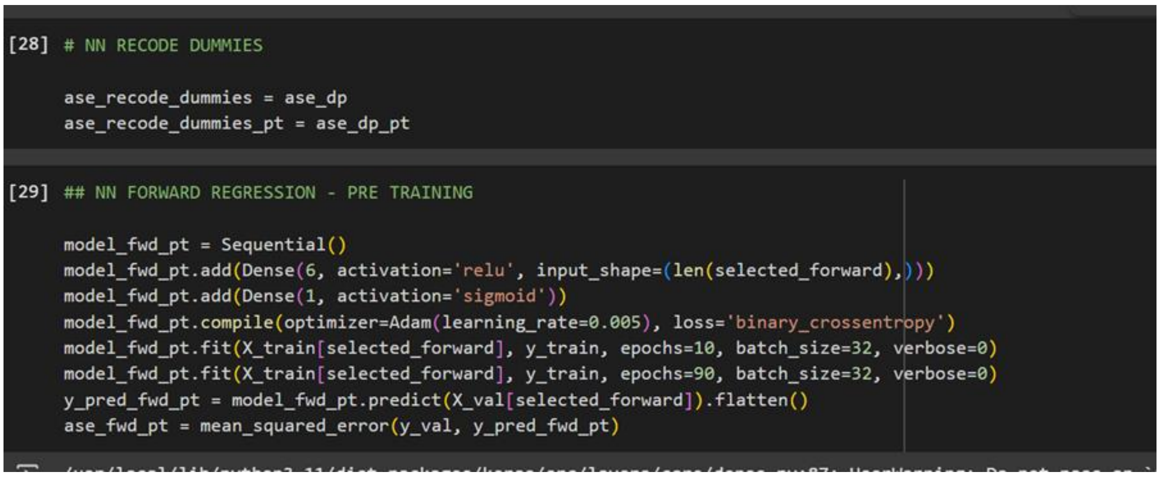
Data partition and pre train



```
[26] # NN DATA PARTITION
     model_dp = Sequential()
     model_dp.add(Dense(6, activation='relu', input_shape=(X_nn_train.shape[1],)))
     model_dp.add(Dense(1, activation='sigmoid'))
     model_dp.compile(optimizer=Adam(learning_rate=0.01), loss='binary_crossentropy')
     model_dp.fit(X_nn_train, y_nn_train, epochs=100, batch_size=32, verbose=0)
     y_pred_dp = model_dp.predict(X_nn_val).flatten()
     ase_dp = mean_squared_error(y_nn_val, y_pred_dp)
```

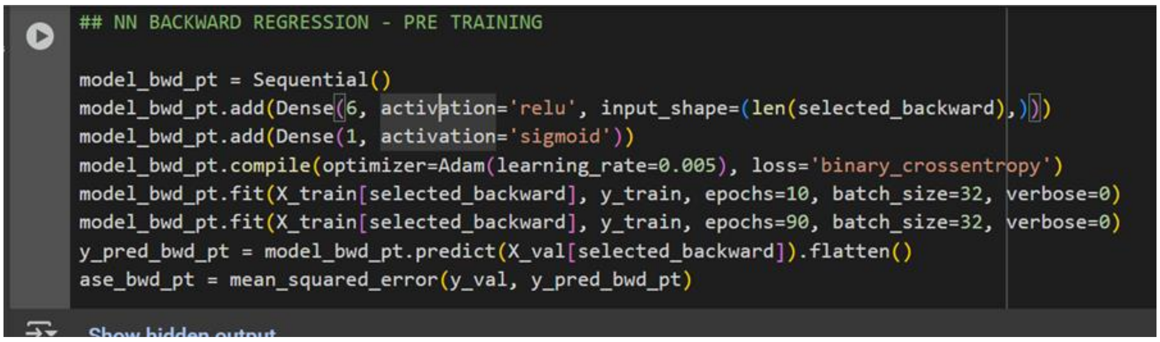
```
## NN DATA PARTITION - PRE TRAINING
model_dp_pt = Sequential()
model_dp_pt.add(Dense(6, activation='relu', input_shape=(X_nn_train.shape[1],)))
model_dp_pt.add(Dense(1, activation='sigmoid'))
model_dp_pt.compile(optimizer=Adam(learning_rate=0.005), loss='binary_crossentropy')
model_dp_pt.fit(X_nn_train, y_nn_train, epochs=10, batch_size=32, verbose=0)
model_dp_pt.fit(X_nn_train, y_nn_train, epochs=90, batch_size=32, verbose=0)
y_pred_dp_pt = model_dp_pt.predict(X_nn_val).flatten()
ase_dp_pt = mean_squared_error(y_nn_val, y_pred_dp_pt)
```

NN forward, backward and stepwise pre-train



```
[28] # NN RECODE DUMMIES

     ase_recode_dummies = ase_dp
     ase_recode_dummies_pt = ase_dp_pt
```

```
[29] ## NN FORWARD REGRESSION - PRE TRAINING

     model_fwd_pt = Sequential()
     model_fwd_pt.add(Dense(6, activation='relu', input_shape=(len(selected_forward),)))
     model_fwd_pt.add(Dense(1, activation='sigmoid'))
     model_fwd_pt.compile(optimizer=Adam(learning_rate=0.005), loss='binary_crossentropy')
     model_fwd_pt.fit(X_train[selected_forward], y_train, epochs=10, batch_size=32, verbose=0)
     model_fwd_pt.fit(X_train[selected_forward], y_train, epochs=90, batch_size=32, verbose=0)
     y_pred_fwd_pt = model_fwd_pt.predict(X_val[selected_forward]).flatten()
     ase_fwd_pt = mean_squared_error(y_val, y_pred_fwd_pt)

     /usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an
```



```
## NN BACKWARD REGRESSION - PRE TRAINING

model_bwd_pt = Sequential()
model_bwd_pt.add(Dense(6, activation='relu', input_shape=(len(selected_backward),)))
model_bwd_pt.add(Dense(1, activation='sigmoid'))
model_bwd_pt.compile(optimizer=Adam(learning_rate=0.005), loss='binary_crossentropy')
model_bwd_pt.fit(X_train[selected_backward], y_train, epochs=10, batch_size=32, verbose=0)
model_bwd_pt.fit(X_train[selected_backward], y_train, epochs=90, batch_size=32, verbose=0)
y_pred_bwd_pt = model_bwd_pt.predict(X_val[selected_backward]).flatten()
ase_bwd_pt = mean_squared_error(y_val, y_pred_bwd_pt)
```

CENTENNIAL COLLEGE