DEEP LEARNING

Main objective of the analysis that specifies whether your model will be focused on a specific type of Deep Learning or Reinforcement Learning algorithm and the benefits that your analysis brings to the business or stakeholders of this data.

Main objective with this work is to focus on Deep Learning for prediction purposes.

Benefits:

- Main aim for this project from a business perspective is to reduce cost the company.
- Context here is that a company is hiring people for sales and training them so a huge upfront cost is involved.
- This model tries to predict during the time of application itself whether an applicant will turn out to be a good sales agent or not.

Brief description of the data set you chose, a summary of its attributes, and an outline of what you are trying to accomplish with this analysis.

The data I took is about hiring new associates/ salespeople by managers in sales division. Since the company is making upfront investment in hiring and training, it tries to predict based on past data if an applicant will produce business after getting hired in future.

So, the data is made up of 14572 rows (each of a distinct applicant) and 22 features for prediction and one binary target variable of which training data has 9527 entries, and test has rest. So we have 14572 people who applied in the past and got hired and whether they sold a product within three months is reflected in business sourced target column. The features/ variables are:

- Identification: 'ID'
- Location of Offices: 'Office_PIN'
- Applicant Details: 'Application_Receipt_Date', 'Applicant_City_PIN', 'Applicant_Gender', 'Applicant_BirthDate', 'Applicant_Marital_Status', 'Applicant_Occupation', 'Applicant_Qualification'
- Manager details: 'Manager_DOJ', 'Manager_Joining_Designation', 'Manager_Current_Designation', 'Manager_Grade', 'Manager_Status', 'Manager_Gender', 'Manager_DoB', 'Manager_Num_Application', 'Manager_Num_Coded', 'Manager_Business', 'Manager_Num_Products', 'Manager_Business2', 'Manager_Num_Products2'
- Target: 'Business_Sourced'[1]

Here, variable types are:

- Date: 'Application_Receipt_Date', 'Applicant_BirthDate', 'Manager_DOJ' and 'Manager_DoB'

[1] For the rest of the document, successful means Business sourced is 1, otherwise 0.

- Numerical: 'Manager_Num_Application', 'Manager_Business', 'Manager_Num_Products', 'Manager_Business2', 'Manager_Num_Products2'

Categorical: rest all

I am trying to model the data to predict whether an applicant now will become successful in future or not, ie, Business sourced is 1 or not.

Brief summary of data exploration and actions taken for data cleaning and feature engineering.

1. Handling date variables
    a. Since we cannot handle date variables directly, we can get the information out of it by creating new columns.
    b. To see if month or year of application date has any effect in determining target, creating separate columns each for month and year instead of date directly
    c. From applicant's DoB, creating a variable for applicant's age since I believe month and date wont have any meaningful impact on their success of work
    d. Similarly using 'Manager_DOJ' and 'Manager_DoB' to calculate manager's age and experience in the company
2. Handling non-numerical variables:
    a. PIN codes were used as geocodes to identify distance between applicant address and office location.
    b. Gender and Manager Status (whether probation or confirmed) were binary encoded.
    c. One Hot encoding (encoding with indicator variables) was performed for: 'Applicant_Marital_Status', 'Applicant Occupation'
    d. Label encoding (Rank ordering) was performed for: 'Manager_Joining_Designation', 'Manager_Current_Designation', 'Manager_Grade'. Grade was already ordinal; designation was of format "Level xx" was converted to ran based on xx.
    e. Target encoding was performed for: 'Applicant_Qualification'
3. Handling missing values
    a. I noticed something in the data wrt missing values, ie, for 683 observations there was no information on any column describing manager detils. So I figured these entries must mean, they had no manage r, So created a new column for indicating the presence of manager.
    b. Since every other variable with missing values except birthdate was a categorical variable and #missi ng is less than half a percent of overall data, observations with birthdate missing were dropped and others excluding Applicant_Occupation were imputed with KNN iteratively with Applicant_Gender h aving least missing imputed first. Due to large number of missing values, Applicant_Occupation missi ng was treated as a new category.
4. Handling outliers
    a. 'Manager_Num_Products' had unproportionally large count of 0 and very low counts of extremely high values. Hence, they were floored and capped at 10, 90 percentiles repectively and indicator for flooring and capping was added as a new feature
    b. 'Manager_Business' also had a couple of negative values and some extremely high values. Hence, they were capped and floored at 95, 5 percentiles respectively.
5. Variable Transformation
    a. 'Manager_Business' was heavily skewed towards left hence a log transformation was applied.
    b. 'Manager_Business2', 'Manager_Num_Products2' are essentially second line of business of a manager if exists. So, when it doesn't the data replicates the first and replicates them which leads to

extremely high correlation between them. So instead of these, their difference from 1st line of business, ie, 'Manager_Business', 'Manager_Num_Products' were calculated and used to create new columns and dropping these.

c. A new column for growth of manager was calculated as a difference of 'Manager_Joining_Designation' & 'Manager_Current_Designation'
d. Applicant_Qualification was re-grouped into Class X, Class XII, Graduate, Certified Associate, Masters & Others and target encoded.

Summary of training at least three variations of the Deep Learning model you selected.

The modelling methods which I took into consideration were: Multi-layer perceptron Classifier with exponentially decreasing node count, Keras neural nets, one with shallow and wide node structure (depth = 2) and one with deep and narrow node structure (depth = 6). Note that all of these we modelled and predicted on same train & test data.

Just to use for comparison, I revisited the previous **IBM course on Supervised learning: Classification**. In that, tree-based methods gave best performance with test and train data among parametric and non-parametric non-black box models. So, I used those to compare with neural net models.

The exact hyper-parameters used for each method were[2]:

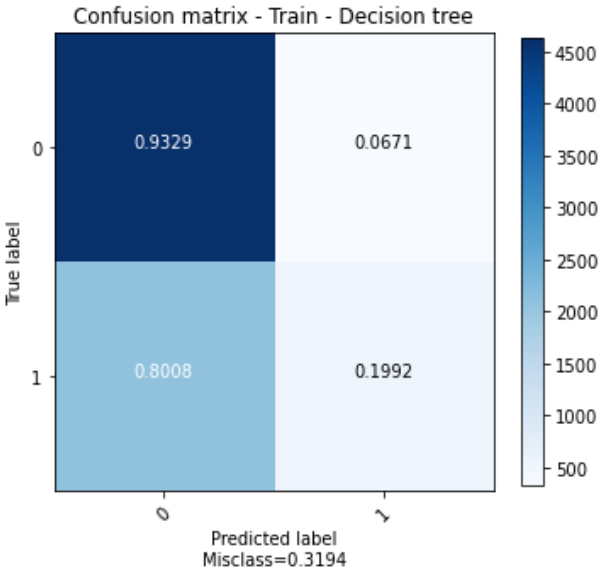
1. ***DecisionTreeClassifier****(max_depth=7, min_samples_split=50, min_samples_leaf=10,min_impurity_decrease=0.0001)*
2. ***RandomForestClassifier****(oob_score=True,max_depth=7, min_samples_split=50,min_samples_leaf=10,min_impurity_decrease=0.0001,n_jobs=-1)*
3. ***MLPClassifier****(hidden_layer_sizes=(1000,100,10,), activation='relu', solver='lbfgs', alpha=0.0001, learning_rate='adaptive', max_iter=150, tol=0.0001, warm_start=True, early_stopping=False, validation_fraction=0.2, beta_1=0.9, beta_2=0.999,epsilon=1e-08, n_iter_no_change=3, max_fun=15000)*
4. ***Shallow Neural Net:*** *model = Sequential()*
    a. *model.add(Dense(5000, input_dim=26, activation='relu'))*
    b. *model.add(Dense(500, activation='relu'))*
    c. *model.add(Dense(1, activation='sigmoid'))*
    d. *model.compile(loss='binary_crossentropy', optimizer='Adamax', metrics=['accuracy'])*
5. ***Deep Neural Net****: Sequential([*
    a. *Dense(2000, activation="relu",kernel_regularizer=l2(1e-4), input_shape=(26,)),*
    b. *Dense(1000, activation="relu"),Dropout(0.3),*
    c. *Dense(500, activation="relu"), Dropout(0.3),*
    d. *Dense(100, activation="relu"), Dropout(0.3),*
    e. *Dense(50, activation="relu"), Dropout(0.3),*
    f. *Dense(10, activation="relu"),*
    g. *Dense(1, activation="sigmoid"),]).compile(loss='binary_crossentropy', optimizer='Adamax', metrics=['accuracy'])*

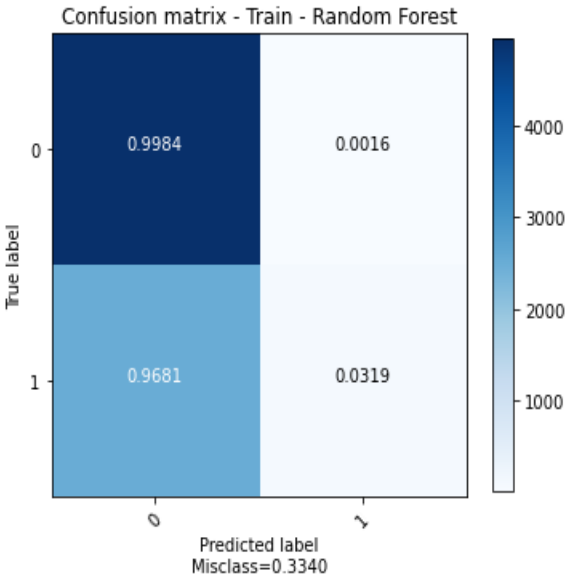
[2] Note that all these hyper parameters have been selected as best after doing Grid Search on various possible combinations.

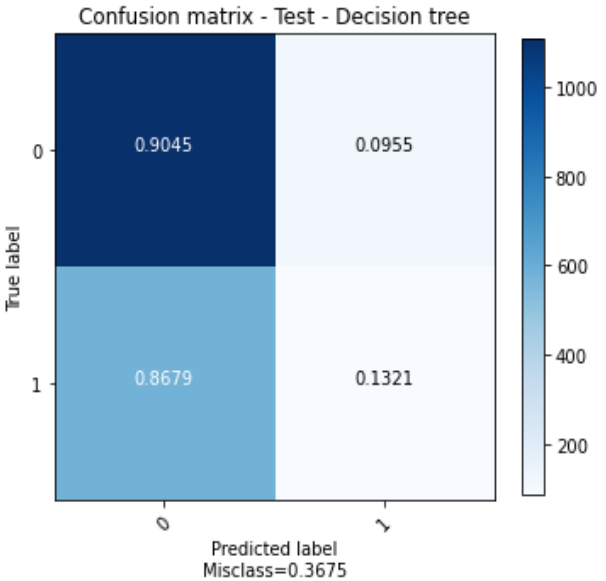
The results I got were:

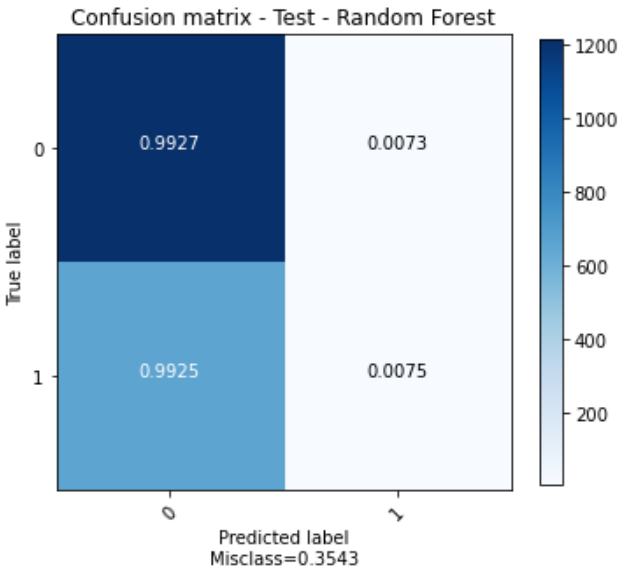
TRAINING DATA						
Method	**Accuracy**	**Cohen's kappa**	**f1 score**	**Precision**	**Recall**	**ROC-AUC score**
Decision Tree	0.68	0.15	0.3	0.61	0.19	0.56
Random Forest	0.66	0.04	0.06	0.91	0.03	0.51
MLP Network	0.68	0.17	0.33	0.6	0.23	0.57
Shallow Neural Net	0.75	0.4	0.56	0.72	0.47	0.69
Deep Neural Net	0.81	0.5	0.86	0.95	0.79	0.75
TESTING DATA						
Method	**Accuracy**	**Cohen's kappa**	**f1 score**	**Precision**	**Recall**	**ROC-AUC score**
Decision Tree	0.63	0.04	0.2	0.43	0.13	0.52
Random Forest	0.64	0.01	0.02	0.36	0.01	0.5
MLP Network	0.64	0.07	0.26	0.44	0.18	0.53
Shallow Neural Net	0.62	0.08	0.33	0.42	0.3	0.54
Deep Neural Net	0.68	0.3	0.74	0.94	0.61	0.6

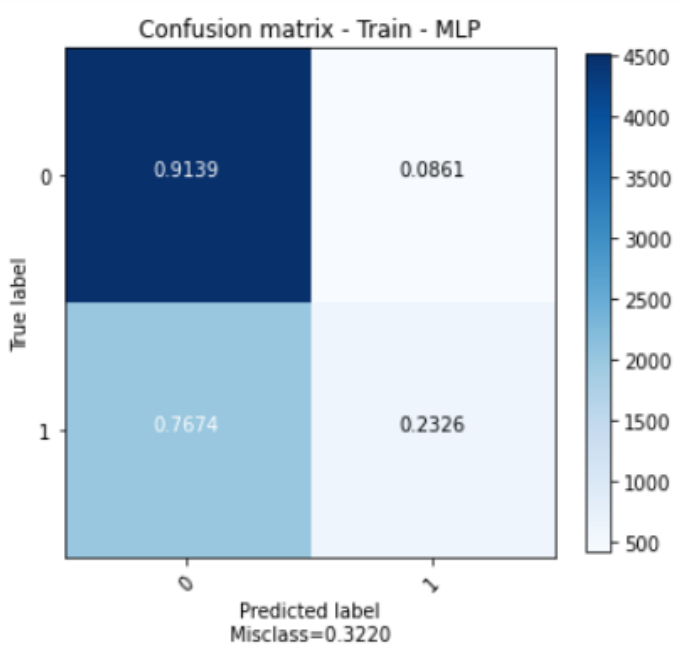
And the confusion matrix for each data sample for each method is shown below:

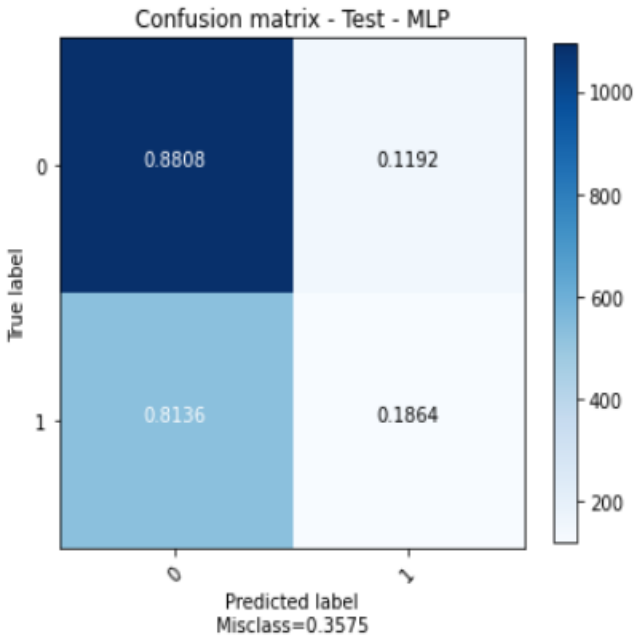


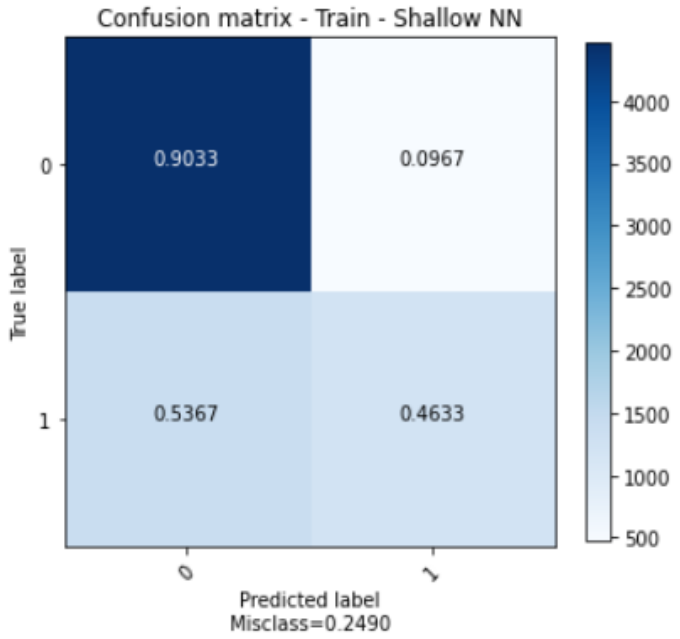


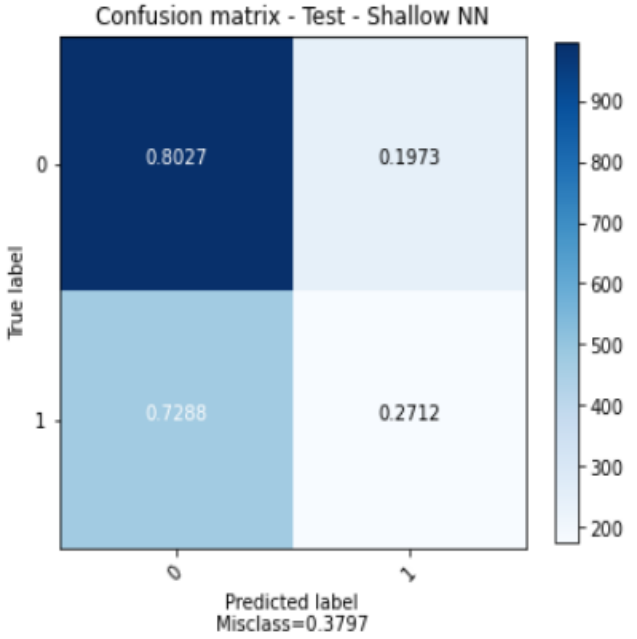


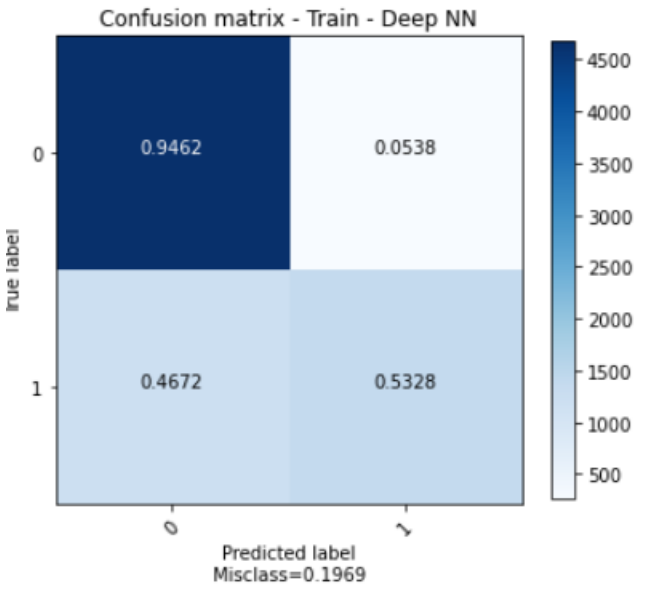


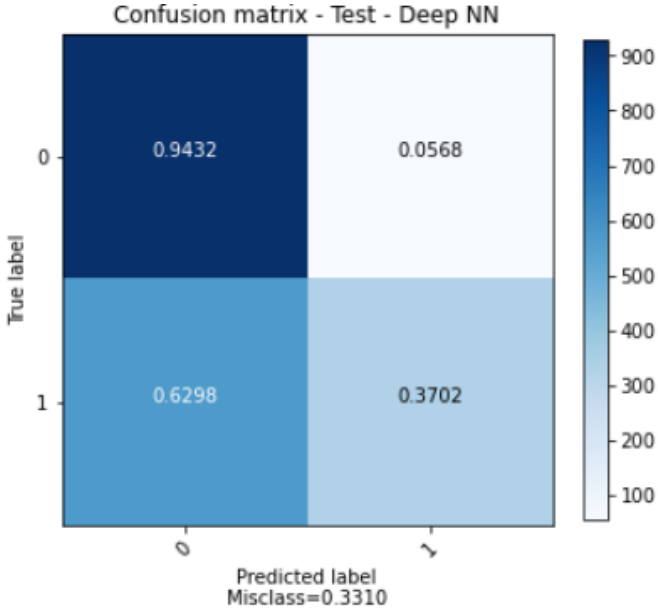

Confusion matrix - Train - MLP
0.9139
0.0861
0.7674
0.2326
True label
Predicted label
Misclass=0.3220


Confusion matrix - Test - MLP
0.8808
0.1192
0.8136
0.1864
True label
Predicted label
Misclass=0.3575


Confusion matrix - Train - Shallow NN
0.9033
0.0967
0.5367
0.4633
True label
Predicted label
Misclass=0.2490


Confusion matrix - Test - Shallow NN
0.8027
0.1973
0.7288
0.2712
True label
Predicted label
Misclass=0.3797


Confusion matrix - Train - Deep NN
0.9462
0.0538
0.4672
0.5328
True label
Predicted label
Misclass=0.1969


Confusion matrix - Test - Deep NN
0.9432
0.0568
0.6298
0.3702
True label
Predicted label
Misclass=0.3310

A paragraph explaining which of your classifier models you recommend as a final model that best fits your needs in terms of accuracy and explainability.

Based on summary statistics of results, we can infer the following:

1. For all the methods, with respect to all comparison metrics, train and test data perform similarly.
2. Neural nets give better performance with both train and test data over tree-based methods.

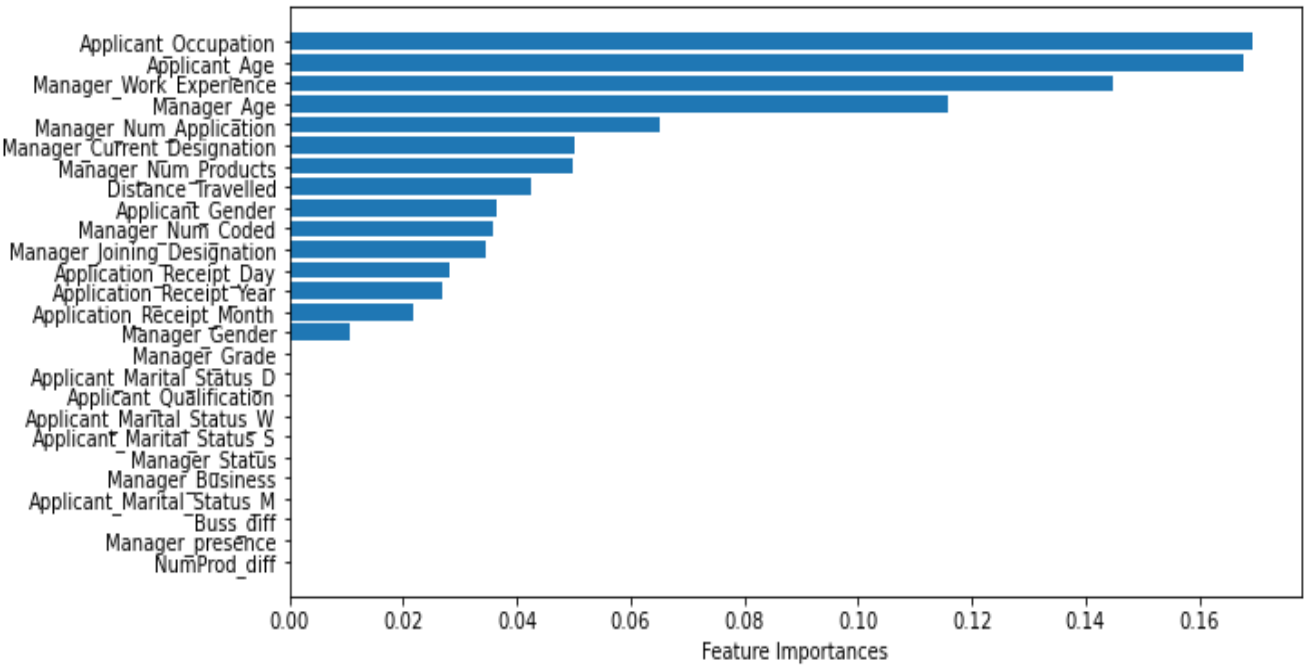
As we can infer from summary stats results, feed forward neural nets perform better than MLP in both accuracy and precision while being almost the same in recall. So, while considering among feedforward NN, shallow NN has more nodes in each layer but fewer layers so they couldn't make accurate prediction in test data due to shortage of classifying options through different layers. But deep NN has good performance in train and test data and due to deep structure could make more complex decision boundary hence giving better predictions.

With respect to explainability all NN models are not good, so we need additional packages like LIME and SHAP to interpret predictions in test data.

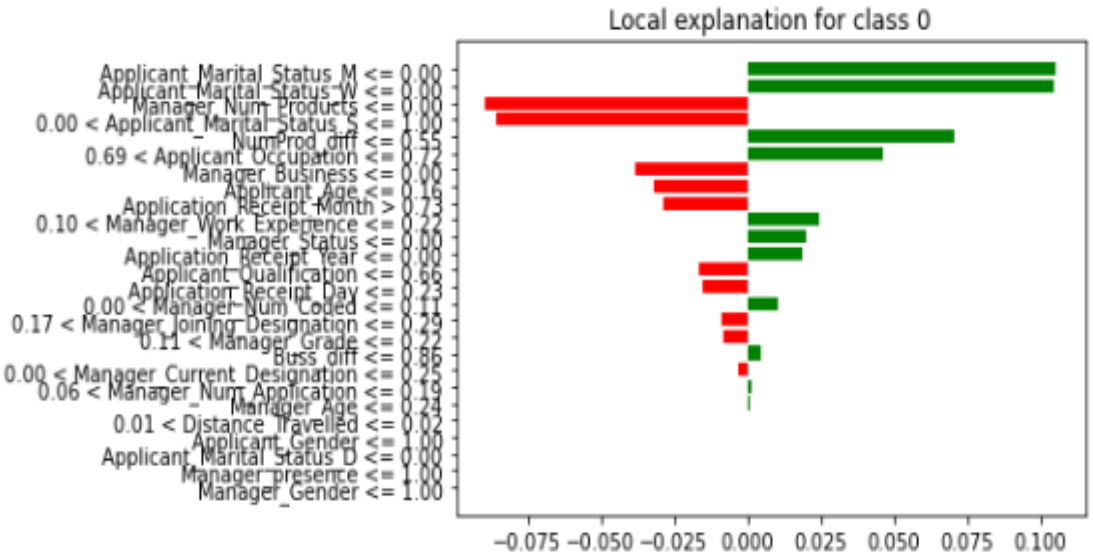
Hence, I will recommend to use Deep neural network model with hyperparameters as mentioned above.

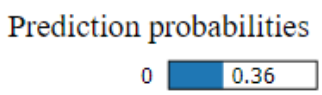
Summary Key Findings and Insights, which walks your reader through the main drivers of your model and insights from your data derived from your classifier model.

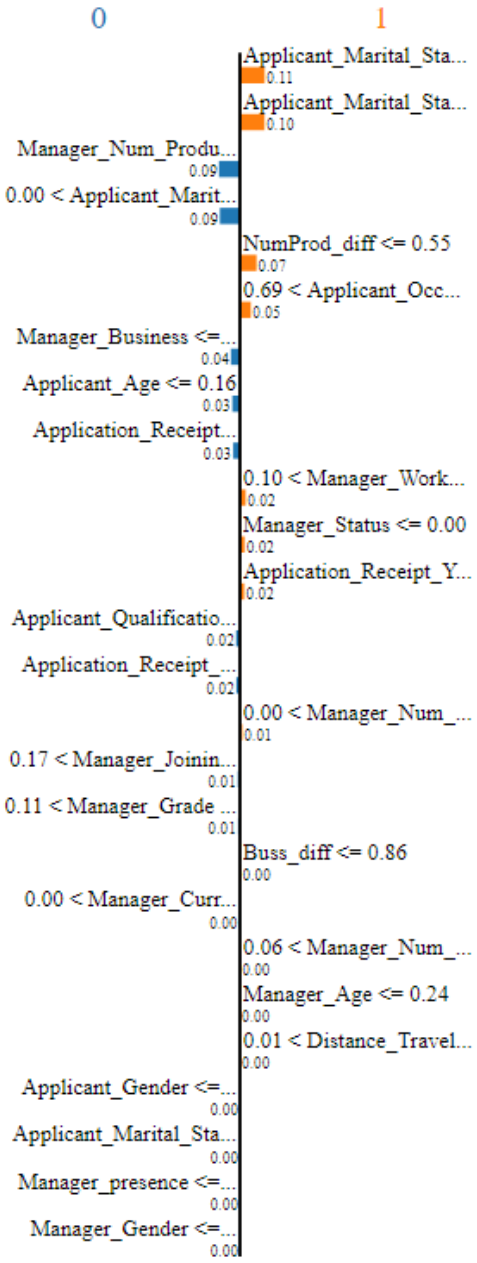
Feature importances based on decision tree is plotted here:

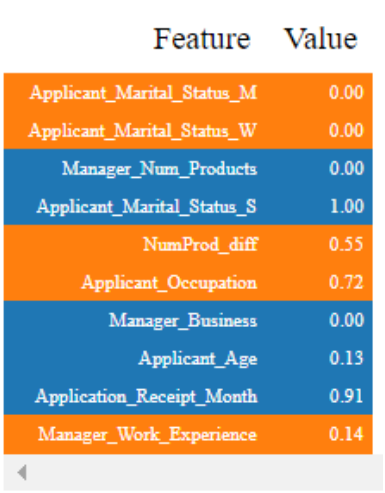


Output of LIME explainer for first observation in test: (This can be used to see what variable is significant is driving the predicted probablity)

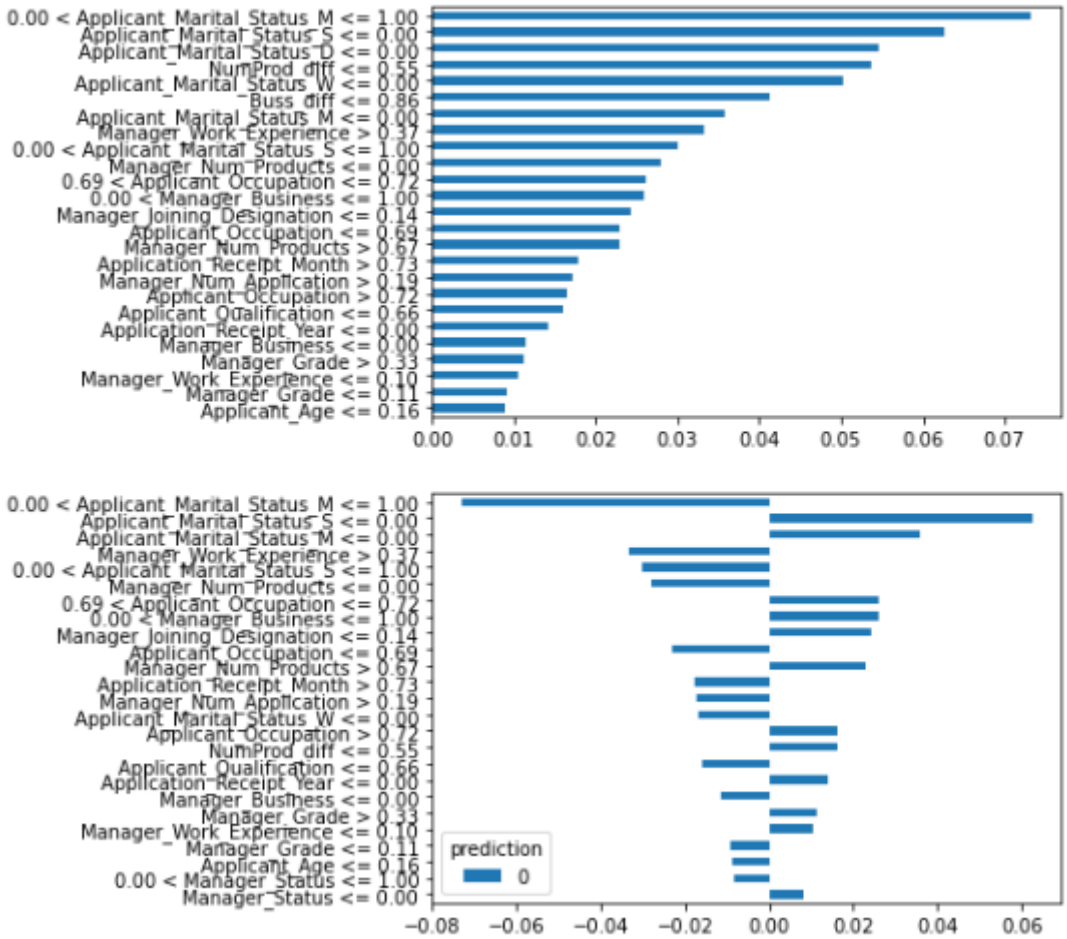








Each instance explanation is agglomerated at global level to give feature split importance both overall and at prediction class level.



As we can infer from above plots, as expected, an applicants' current occupation, Manager's work experience, number of applications they received come out as very important factors in determining success of an applicant. What's more interesting and counter intuitive is Applicant's marital status is also an important driver.

Suggestions for next steps in analysing this data, which may include suggesting revisiting this model after adding specific data features that may help you achieve a better explanation or a better prediction.

Immediate next step would be to investigate why marital status is coming out as an important driver. Other next steps would be to do further Hyper parameter tuning probably randomised or Bayesian to improve its overall accuracy and precision. Also, next step is to do model interpretability analyses on SHAP to get better insights on global level feature interaction and dependence plots.

Additional data regarding performance of applicant in their occupation would give better insights into the overall capacity of the applicant. Also, data on whether that applicant has previous experience in sales would also be a good predictor I believe. Also, if more data overall were available, it would be great since more the data usually better the NN model.